Zero knowledge from privacy-free garbled circuits

glaroton

November 16, 2026

Chapter 1

Introduction

This blueprint describes the mathematics formalized in the Lean library gc-zk: zero-knowledge proofs built from privacy-free garbled circuits. Every definition and theorem below names its Lean counterpart, and the web version of this document has a dependency graph that shows how the results build on each other. All the statements in this document are proved in Lean, with no `sorry`, and the main theorems depend only on the standard axioms of Lean.

Zero knowledge from garbled circuits

The idea of a garbled circuit is due to Yao, who presented it orally in the context of secure two-party computation; the construction does not appear in his written paper [Yao86], and its history is told by Bellare, Hoang and Rogaway [BHR12b, Sect. 1]. In 2013 Jawurek, Kerschbaum and Orlandi [JKO13] observed that they also give an efficient zero-knowledge proof for any statement of the form "I know w with $C(w) = 1$" for a Boolean circuit C. The roles are reversed: the verifier garbles C, the prover evaluates it on its witness, and the garbling is opened at the end so that the prover can check that it was honest. Since the prover knows its own input, the garbling does not need to hide anything from the evaluator, only to prevent it from forging output labels. Frederiksen, Nielsen and Orlandi called such schemes *privacy-free* [FNO15, Sect. 2.2, Definition 4] and showed that they are considerably cheaper; their adaptation of Free XOR garbles each non-XOR gate with two ciphertexts [FNO15, Sect. 1.2]. With Free XOR [KS08] and the half-gates of Zahur, Rosulek and Evans, XOR gates cost nothing and AND gates cost one ciphertext [ZRE15, Sect. 6].

The garbling scheme formalized here is the privacy-free scheme of Zahur, Rosulek and Evans [ZRE15, Sect. 6]: Free XOR, and for the non-linear gate their one-ciphertext evaluator half-gate [ZRE15, Sect. 3.1], written as a multiplexer (Remark 3.5). Circuits are built from inputs, constants, XOR gates and multiplexers.

What the protocol is for

The protocol is an interactive zero-knowledge argument of knowledge (soundness holds against provers with bounded resources) for statements of the form "I know w with $C(w) = 1$", where C is any Boolean circuit known to both parties. Its cost is proportional to the size of C [JKO13, Sect. 1], and it uses only symmetric primitives (a hash) besides the oblivious transfers, one per

bit of the witness. It is therefore suited to statements without algebraic structure, for which the classical zero-knowledge protocols for discrete logarithms or other group relations do not apply.

Typical uses. The examples of Jawurek, Kerschbaum and Orlandi [JKO13, Sect. 1 and 1.1] are of this kind:

- knowledge of a preimage, "I know x with $y = \text{SHA-256}(x)$" for a public y;
- knowledge of a key, "I know k with $y_1 = \text{AES}_k(y_2)$";
- statements about data encrypted with a block cipher, for instance that part of a ciphertext decrypts to a given value, without releasing the key;
- knowledge of a signature, or of a Fiat–Shamir proof, without revealing it, which requires proving statements about the hash function that instantiates the random oracle.

They report experiments for AES, MD5 and SHA-256 [JKO13, Sect. 5.2]. With the privacy-free half-gates garbling formalized here, the garbled circuit for "I know k such that $\text{AES}(k, m) = c$" takes about 108 kilobytes with 128-bit labels [ZRE15, Sect. 6].

What it does not give.

- *It is interactive, and the proof convinces only the verifier.* The verifier's messages are not just random challenges, so the protocol is not public-coin and the Fiat–Shamir transformation does not apply [JKO13, Sect. 3, footnote 6]. The protocol as formalized here has no non-interactive or publicly verifiable version: the verifier itself builds the garbled circuit and later opens it, so a transcript is not evidence for anyone else. It does not fit settings where a proof must be posted once and checked by many.
- *One garbling per proof.* At the end the verifier opens its seed, which reveals the global offset Δ, so a garbled circuit cannot be reused for another proof.

Ways to make it insecure. Each of the following breaks a hypothesis of the theorems below, or makes their bounds useless.

- *Skipping the prover's check.* A malicious verifier can garble a circuit other than C, or mount a selective-failure attack on the input phase [JKO13, Sect. 1]. Zero knowledge rests on the prover committing to its output label, verifying the opened garbling against C and the labels it received, and only then revealing the label (Theorem 4.5). A prover that reveals the label before the check, or that does not check, leaks information about its witness; Proposition 4.10 gives an explicit example.
- *Sending the garbled circuit before the oblivious transfers, or using an OT that lets the prover choose its input late.* If the circuit is sent before the oblivious transfers, the garbling must be secure against adaptive attacks [JKO13, Sect. 3, footnote 7]. The same holds whenever the prover can choose its witness bits after seeing the garbled circuit. This is the situation of the Chou–Orlandi OT, where a receiver can delay its decryption; it is treated in Chapters 10 and 11.
- *Weak oblivious transfer keys.* With the PVW OT, a receiver key $(0, 0)$ makes every ciphertext carry its message in the clear; the sender in this library rejects every key whose first component is 0 (Lemma 9.4). With the Chou–Orlandi OT, a secret $a_j = 0$ makes the two masks of input j equal; this costs k/q in the soundness bound (Theorem 11.2).

- *An unsuitable hash.* Correlation robustness of the hash alone is not enough to prove Free XOR secure: Choi, Katz, Kumaresan and Zhou rule out a fully black-box reduction [CKKZ12, Sect. 3]. The theorems here assume a random oracle; for Free XOR in Yao's protocol a circular form of correlation robustness suffices [CKKZ12, Sect. 4].
- *Small parameters.* The bounds below are of order $q/|V|$ for q hash queries; the labels must be long enough (for instance 128 bits) for this to be negligible, and likewise the commitment values and, with a real oblivious transfer, the order of the group.

No claim of originality

No originality is claimed for any definition, statement or proof idea in this library or in this document. The protocols, the security notions and the arguments come from the cited literature, or are standard in it; what is ours is only their formalization in Lean. Where we could not yet match a statement with a published one, we say so. Should any of them turn out to be new, this will be stated separately.

Contents

Chapter 2 explains how the order of the messages (when the witness is fixed, when the garbling is opened) and the choice of commitment and oblivious transfer determine the security that can be proved, discusses security against adaptive corruptions, and points to the theorems that cover each case. Chapter 3 defines circuits, the garbling and the evaluation, and proves correctness. Chapter 4 formalizes the protocol of [JKO13] in the hybrid model with ideal oblivious transfer and commitment, and proves completeness, perfect zero knowledge against a malicious verifier, and soundness by reduction to authenticity. Chapter 5 shows that Δ reaches the evaluator only through the hashes of inactive labels, and Chapter 6 bounds the authenticity advantage of a q-query adversary in the random oracle model by $(q+1)/(|V|-1)$, for the real garbling with uniform zero-labels. Chapter 7 replaces the ideal commitment by a hash commitment in the random oracle model, with statistical zero knowledge and soundness by extraction. Chapter 8 replaces the ideal OT by the Chou–Orlandi OT and proves zero knowledge with both real components. Chapter 9 uses instead the dual-mode OT of Peikert, Vaikuntanathan and Waters [PVW08], which realizes the OT functionality of [JKO13]: soundness holds in messy mode, and zero knowledge is reduced to DDH by explicit distinguishers. With the Chou–Orlandi OT the prover can choose its input after seeing the garbled circuit, so soundness needs *adaptive* authenticity: Chapter 10 proves that an adversary that interleaves hash queries and label requests wins with probability at most $(q+1)/(|V|-1) + q/|V|$, and Chapter 11 uses it to prove soundness with the Chou–Orlandi OT, by reduction to the square Diffie–Hellman problem. Chapter 12 proves some ingredients for adaptive corruptions: the equivocation condition for the receivers of the oblivious transfers, the explanation of the prover's messages for any witness, the equivocation of the hash commitment, and soundness with the PVW OT in decryption mode. Chapter 13 contains the cut-and-choose bounds against a malicious garbler, derived from the garbling, and Chapter 14 lists what is not formalized.

The model

All the security statements are information-theoretic statements in idealized models (the random oracle model [BR93], and the hybrid models of [JKO13]), or reductions to explicit adversaries

against DDH and square Diffie–Hellman. They are proved by counting: a probability is the proportion of the randomness on which an event holds, and the proofs are sequences of games [Sho04] in which "identical until bad" [BR06] becomes an injection between sets of randomness. Adversaries are deterministic; a randomized adversary is a family of them indexed by its coins, and every bound holds for each member of the family, hence on average. The numbers of queries are fixed; an adversary that makes fewer queries, or interleaves its queries to several oracles differently, is brought to this form by padding with dummy queries, an argument that is stated but not formalized. Lean has no notion of efficient computation, so polynomial-time constraints, and the efficiency of the reductions, are outside the formalization.

Notation. Labels live in a vector space V over $\mathbb{F}_2$, and $\Delta \in V$ is the global offset. The letter q has two uses: in Chapters 6 and 10 it is the number of queries (or actions) of the adversary; in Chapters 8, 9, 11 and 12 it is the order of the group $\mathbb{G}$ with generator g, and numbers of queries are written $q_0, \ldots, q_3$, Q, n_0, n_1, n_2 or N_1, N_2. Chapter 2, which summarizes all of them, uses q in both senses, each time as in the chapter it refers to. From Chapter 7 on, G is the commitment oracle, distinct from the group $\mathbb{G}$. A circuit input is x (or the witness w), and b is a bit, except in the oblivious transfers, where b is the receiver's secret exponent. In Chapter 9, b is also the branch of the cryptosystem, and x_0, x_1, x, y are exponents of the common string.

Related work

Formalized cryptography. Game-based cryptographic proofs have been mechanized in several frameworks, among them EasyCrypt (see [ABB+17]), CryptHOL in Isabelle/HOL [BLS20], SSProve in Coq [HRV+23], and, in Lean 4, VCVio [TDW+26] and CatCrypt [Spi26]. Closest to the present library are the EasyCrypt verification of Yao's secure function evaluation protocol, with a formalization of garbled circuits in the framework of Bellare, Hoang and Rogaway [ABB+17, BHR12b], and the formalization of simulation-based security of multi-party computation in Isabelle/HOL by Butler, Aspinall and Gascón [BAG17]. In Lean 4, Dziembowski, Fabiański, Micciancio and Stefański formalize the computational soundness of a symbolic framework and, as an extended example, the simulation-based privacy of a garbling scheme taken from Li and Micciancio, proved symbolically [DFMS26, Sect. 3]. Among its 172 protocols, CatCrypt lists Yao's garbled circuits with half-gates, row reduction and Free XOR, and a DDH-based "Simplest OT" [Spi26, Sect. 4]; its protocol sources are in a private repository [Spi26, Sect. 7] (only the core is public), so we have not compared its definitions or bounds with ours. As of November 2026, the README of the public core also lists machinery for universal composability: UC emulation, the dummy-adversary theorem and a composition law, and says that the universal-composability theory itself is in separate distributions [CCcore]. We are not aware of a previous formalization of zero knowledge from garbled circuits or of privacy-free garbling. The present library does not use a general framework: its probabilities are counts over finite types, and its games are written directly as functions of the randomness.

Adaptive garbling. Adaptive security of garbling, where the input is chosen after the garbled circuit, was introduced by Bellare, Hoang and Rogaway [BHR12], who define adaptive privacy [BHR12, Sect. 3, Fig. 2] and adaptive obliviousness and authenticity [BHR12, Sect. 4, Fig. 6], and give transforms that make any statically secure scheme adaptively secure. Hemenway, Jafargholi, Ostrovsky, Scafuro and Wichs prove simulation-based adaptive security, a privacy notion, for a modification of Yao's scheme under one-way functions [HJO+16, Sect. 3 and 5]; Jafargholi and Wichs prove the same notion for Yao's scheme itself, unmodified, with a security loss governed by a pebbling game [JW16, Sect. 3 and Theorem 1]. Adaptive security of unmodified half-gates was studied in idealized models by Barnum, Heath, Kolesnikov and Ostrovsky [BHKO26] and by

Guo, Yang, Wang, Yu and Liu [GYWYL]. Chapter 10 needs adaptive authenticity of unmodified privacy-free garbling, for which we have not found a published statement. Ganesh, Kondi, Patra and Sarkar [GKPS18, Sect. 3] show that the protocol of [JKO13] is secure against *adaptive corruptions* when the oblivious transfer satisfies a mild adaptive security guarantee; this is a different notion from the adaptive choice of the input considered here.

The Chou–Orlandi OT. The issue that motivates Chapters 10 and 11, that a receiver can delay the decryption, was pointed out by Byali, Patra, Ravi and Sarkar [BPRS17, Sect. 3] and discussed by Doerner, Kondi, Lee and shelat [DKLs18, App. A], who add a verification step to the protocol; Chou and Orlandi refer to both [CO15, Sect. 1.1] and withdraw the claim of UC security of their protocol.

How this library was written

The author chose the protocol, the model and the statements, and reviewed the results. The Lean proofs were written by the AI assistant Claude (Anthropic) working under his direction, and so was a first draft of the pen-and-paper arguments of Chapters 10 and 11 (the linear parts of labels, the bad steps, and the reduction with guessed key queries), which the author then reviewed. Correctness does not depend on trusting either of them: Lean checks every proof. The formalization uses Lean 4 [MU21] and Mathlib [Mathlib20], and this document is built with leanblueprint [Mas].

Chapter 2

Message order and the security it gives

What can be proved about the protocol of [JKO13] depends less on the garbling scheme than on the order of the messages: when the prover's witness bits become fixed, compared with the moment it sees the garbled circuit, and when the verifier opens its garbling, compared with the moment the prover discloses its output label. The literature fixes these choices in different places, often only in a figure or a footnote. This chapter collects them, says which kind of security each choice allows, and points to the statements of this library that cover it. The chapter is expository and is not itself formalized. It contains no new statement; every claim about the library refers to a theorem proved in a later chapter, and a phrase such as "soundness needs adaptive authenticity" describes what the proofs here use, not a proved necessity.

2.1 The protocol in order

In the protocol of Jawurek, Kerschbaum and Orlandi [JKO13, Sect. 1.1, Fig. 1] the messages are, in order:

1. the prover runs the oblivious transfers as receiver, one per bit of its witness w; the verifier, as sender, inputs the two labels of each input wire;
2. the verifier sends the garbled circuit (with a two-round OT, together with its second message);
3. the prover evaluates and *commits* to the output label it obtained;
4. the verifier *opens* its garbling: the seed, and all its inputs to the oblivious transfers ("open-all");
5. the prover checks that the opened garbling is a garbling of the agreed circuit and agrees with the labels it received, and only then *reveals* its committed label;
6. the verifier accepts if the label is the output label for the value 1.

Three choices determine the security: the position of step 2 with respect to the moment the witness bits are fixed in step 1; whether step 4 exists and comes after step 3; and how the commitment and the oblivious transfer are realized.

2.2 When the witness bits are fixed: static or adaptive authenticity

Witness fixed before the garbled circuit is seen. If the receiver's message in the OT determines its choice bits, and the garbled circuit is sent only afterwards, the prover's input does not depend on the garbling. Soundness then needs only the static authenticity of the garbling: the input is chosen first, and the adversary must forge the other output label. This is the order of [JKO13], who note that the garbled circuit may travel with the second message of a two-round OT [JKO13, Sect. 1.1], and that sending it before the oblivious transfers would require a garbling secure against adaptive attacks [JKO13, Sect. 3, footnote 7]. In this library: soundness reduces exactly to the authenticity game (Theorem 4.8), bounded by $(q+1)/(|V|-1)$ in the random oracle model (Theorems 6.9 and 6.12). The PVW OT in messy mode is of this kind: the receiver's key determines its choice, which a trapdoor extracts (Lemma 9.4, Theorem 9.9).

Witness fixed after the garbled circuit is seen. If the garbled circuit is sent before the oblivious transfers, or if the OT lets the receiver postpone its choice, the prover may choose its witness bits as a function of the garbled circuit. Soundness then needs *adaptive* authenticity, in which the adversary sees the garbled circuit and requests its input labels one at a time [BHR12, Sect. 4, Fig. 6]. The Chou–Orlandi OT is of this kind: the receiver's message does not fix its choice, and a receiver can delay the decryption until after it has seen the garbled circuit ([BPRS17, Sect. 3], [DKLs18, App. A]; see also [CO15, Sect. 1.1]). In this library: adaptive authenticity for the garbling formalized here, with bound $(q+1)/(|V|-1)+q/|V|$ for q actions (Theorem 10.17), and soundness with the Chou–Orlandi OT, by reduction to square Diffie–Hellman (Theorem 11.2).

2.3 When the garbling is opened: honest or malicious verifier

No opening. If the prover sends its output label directly and the verifier never opens the garbling, the protocol is zero knowledge only against an honest verifier [JKO13, Sect. 3]: a malicious verifier can garble a circuit other than the agreed one, or mount a selective-failure attack in the input phase [JKO13, Sect. 1]. In this library: honest-verifier zero knowledge (Theorem 4.9), and an explicit malicious verifier that learns a witness bit from the OT pairs alone (Proposition 4.10).

Commit, open, check, reveal. With steps 3 to 5 in this order the protocol is zero knowledge against a malicious verifier, with a single garbled circuit. The opening is possible because in zero knowledge the verifier has no secret input, unlike in general two-party computation [JKO13, Sect. 1]. The order matters: the prover must commit to its label before the opening, check the opening, and reveal the label only afterwards. The OT must allow this opening, so that the prover can check that the labels it received are those of the opened garbling [JKO13, Sect. 1.1]. In this library: perfect zero knowledge in the hybrid model (Theorem 4.5).

The garbling cannot be opened. When the garbler has inputs or keys that must stay secret, the garbling cannot be opened at the end, and a malicious garbler is handled by cut-and-choose: several garbled circuits, some opened and some evaluated. If one good evaluated circuit suffices, choosing each circuit with an independent fair coin gives error 2^{-N} [Lin13]. In this library: Theorems 13.3, 13.4 and 13.9.

2.4 How the commitment and the oblivious transfer are realized

Ideal functionalities. With ideal OT and ideal commitment, zero knowledge is perfect and soundness is exactly the authenticity game (Theorems 4.5 and 4.8).

Hash commitment in the random oracle model. Zero knowledge becomes statistical, with error at most $(q_1 + q_2 + q_3)/|R|$ for a verifier making $q_1 + q_2 + q_3$ oracle queries; the simulator *programs* the oracle (Theorem 7.7). Soundness goes through an extractor that reads the prover's queries, at an extra cost $(q_1^2 + q_2 + 1)/|D|$ for q_1 queries before the commitment and q_2 after it (n_1 and n_2 in Theorem 7.15) (Theorems 7.11 and 7.15).

PVW OT with a common string. In messy mode the receiver's choice is extractable, and soundness keeps the bound of the hybrid model (Theorem 9.9). Zero knowledge is computational: it is reduced to DDH by two explicit distinguishers between the modes (Theorems 9.16 and 9.20). The receiver key $(0, 0)$ sends the messages in the clear, and the sender in this library rejects every key whose first component is 0 (Lemma 9.4).

Chou–Orlandi OT in the random oracle model, without common string. Receiver privacy is perfect and zero knowledge holds with the real OT and the hash commitment (Theorem 8.7). Soundness needs adaptive authenticity, and costs k/q for a secret $a_j = 0$ plus the advantages of explicit square Diffie–Hellman solvers (Theorem 11.2). Chou and Orlandi withdraw the claim of UC security of their protocol [CO15, Sect. 1.1].

2.5 When parties are corrupted

The security theorems of this library assume *static* corruptions: the adversary decides at the start which party it controls. Chapter 12 proves some ingredients for adaptive corruptions, but no security statement against them. In the framework of universal composability (UC) party corruption is modelled by conventions on top of the basic model of execution [Can20, Sect. 7.1]. The formulation given there captures *adaptive* corruptions: the adversary corrupts parties as the computation proceeds, based on what it has seen, and receives the entire local state of a corrupted party. Static corruptions, where the corrupted parties are fixed before the computation starts, are a weaker threat model, and the setting where erasures are not trusted corresponds to protocols that write each data cell at most once [Can20, Sect. 7.1.1]. Against adaptive corruptions a simulator must produce the transcript without the witness and, when the prover is corrupted later, *explain* it: produce internal randomness consistent with the actual witness [GKPS18, Sect. 1.2].

Why it matters with two parties. We add a remark of our own. In a single proof between two parties the verifier has no secret input, and a corrupted prover hands over its witness anyway. Adaptive security matters most when the proof is a component of a larger protocol, with many sessions or many parties, in which devices may be compromised at different times: the security proof of the larger protocol, through the composition theorem [Can20, Theorem 22], then relies on every component being simulatable under late corruption. Ganesh, Kondi, Patra and Sarkar motivate adaptive security by adversaries that use their resources gradually, and by applications that use the protocol of [JKO13] as a component [GKPS18, Sect. 1.3].

What has to be explained, and when. In the protocol of [JKO13] the verifier has no input, so its state never needs to be faked, and once the prover holds its output label nothing it does depends on the witness [GKPS18, Sect. 3.2]; in this library the revealed label is the simulator's output (Theorem 4.5). What must be explained therefore depends on the moment the prover is corrupted. The table is our reading of the protocol; [GKPS18] prove the case of an ideal

commitment, stated after it.

Prover corrupted	**To be explained**	**Property used**
before the oblivious transfers	nothing	none
after the oblivious transfers, before the commitment	its OT messages, for the actual witness	receiver-equivocal OT
after the commitment, before the opening	its OT messages and its commitment	also an equivocal commitment
after the label is revealed	its OT messages; the commitment is already opened	receiver-equivocal OT; the label does not depend on the witness

A receiver-equivocal OT is a secure OT whose receiver message can be produced, from the trapdoor of the setup, without the choice bit and later opened to either bit [GKPS18, Sect. 3, Definition 3.1]; the PVW OT in decryption mode is one, assuming DDH [GKPS18, Sect. 3.3, Theorem 3.2]. DDH is used for the security of the OT; the equivocation itself is exact (Theorem 12.2). With an ideal commitment there is nothing to explain. A hash commitment in the *programmable* random oracle model can be equivocated by setting the oracle at the committed point after the fact, a technique recalled in [GKPS18, Sect. 1.2]. With a receiver-equivocal OT and an ideal commitment, the protocol of [JKO13] is secure against adaptive corruptions [GKPS18, Sect. 3]; after the execution, the only work is to equivocate the prover's view of the OT [GKPS18, Sect. 3.2].

The other choices. The order of the messages interacts with the corruption time. For the Chou–Orlandi OT we note the following, which follows from the form of the messages: the receiver message $B = b \cdot g + wA$, with $A = a \cdot g$, is also $(b + (2w-1)a) \cdot g + (1-w)A$, so explaining it for the other choice bit requires the sender's secret a. The simulator knows a when the verifier is honest, and otherwise learns it only at the open-all step, so a corruption of the prover before the opening is the delicate case. If the garbler has an input of its own, as in general two-party computation, the garbled circuit itself must be equivocated, which requires much heavier tools [GKPS18, Sect. 3.2]. Without erasures there is more to explain, and adaptively secure oblivious transfer is harder to build [BPRS17].

In this library. Adaptive security itself is not formalized: neither the UC framework, nor a model of execution with corruptions. Chapter 12 proves the protocol-specific facts behind the table: the receiver of the PVW OT in decryption mode, with the trapdoor of the setup, and the receiver of the Chou–Orlandi OT, with the sender's secret, satisfy the equivocation condition on messages and randomness (Theorem 12.2, Proposition 12.4); under that condition the prover's OT messages, and the verifier's view of the prover, can be explained for any witness (Theorem 12.5); the hash commitment can be equivocated by programming the oracle (Theorem 12.6); and soundness holds with the PVW OT in decryption mode, up to the DDH advantage (Theorem 12.7). These are ingredients only; they are not composed into a simulation. A somewhat restricted variant of the UC framework has been mechanized in EasyCrypt by Canetti, Stoughton and Varia, with a worked example [CSV19]. The README of the public core of CatCrypt lists machinery for universal composability in Lean, and says that the universal-composability theory itself is in separate distributions [CCcore].

2.6 Summary

Choice	**Security that can be expected**	**In this library**
Witness fixed, then garbled circuit	Soundness from static authenticity	Theorems 4.8, 6.12, 7.15, 9.9
Garbled circuit first, or OT with late choice	Soundness needs adaptive authenticity	Theorems 10.17, 11.2
No opening	Honest-verifier zero knowledge only	Theorem 4.9, Proposition 4.10
Commit, open, check, reveal	Zero knowledge against a malicious verifier, one circuit	Theorems 4.5, 7.7, 8.7, 9.20
Garbling cannot be opened	Cut-and-choose, error 2^{-N}	Theorems 13.4, 13.9
Ideal OT and commitment	Perfect zero knowledge	Theorem 4.5
Hash commitment (random oracle)	Statistical zero knowledge, programmable oracle	Theorem 7.7
PVW OT, messy mode	Computational zero knowledge (DDH)	Theorem 9.20
Chou–Orlandi OT	Soundness by reduction to square Diffie–Hellman	Theorem 11.2
Adaptive corruptions	A receiver-equivocal OT with an ideal commitment suffices [GKPS18]	Some ingredients: Theorems 12.2, 12.5, 12.6, 12.7; adaptive security itself is not formalized (Section 2.5)

Security against adaptive corruptions, a different axis, is discussed in Section 2.5.

Chapter 3

Privacy-free Free-XOR garbling

Labels live in a vector space V over $\mathbb{F}_2$; for instance $V = \mathbb{F}_2^{128}$, that is, 128-bit strings with XOR. A wire carries one of two labels, L for the bit 0 and $L + \Delta$ for the bit 1, where $\Delta \in V$ is a global offset chosen by the garbler. The evaluator holds one of them, the *active* label; the other one is *inactive*.

3.1 Circuits and plaintext evaluation

Definition 3.1 (Circuits)**.** A node is one of $\mathsf{input}(j)$, $\mathsf{const}(b)$, $\mathsf{xor}(a, b)$ and $\mathsf{mux}(s, z, o)$, where $j, a, b, s, z, o \in \mathbb{N}$ and $b \in \{0, 1\}$. A circuit is a list of nodes. The operands are positions of earlier nodes; a position that does not refer to an earlier node reads the default wire (value 0, label 0).

The multiplexer $\mathsf{mux}(s, z, o)$ outputs the value of o if s is 1 and the value of z otherwise; AND and OR are multiplexers with a constant candidate. Accepting arbitrary positions avoids carrying a well-formedness hypothesis through every statement.

Definition 3.2 (Plaintext evaluation)**.** For an input $x \colon \mathbb{N} \to \{0, 1\}$, the values of the nodes of C are computed in order: x_j, b, $v_a \oplus v_b$, and v_o or v_z according to v_s.

3.2 Garbling and evaluation

Definition 3.3 (Labels of a bit)**.** For a zero-label L and a bit b, $\mathrm{lab}_\Delta(L, b) = L + b\,\Delta$.

Definition 3.4 (Garbling)**.** The garbler's seed consists of Δ and the zero-labels of the inputs and of the constants; the hash is a map $H \colon \mathbb{N} \times V \to V$ (gate position, label). The node at position i receives a zero-label L_i and publishes some material:

- $\mathsf{input}(j)$: L_i is the zero-label of input j; no material.
- $\mathsf{const}(b)$: L_i is the zero-label of the constant; the material is its active label $\mathrm{lab}_\Delta(L_i, b)$.
- $\mathsf{xor}(a, b)$: $L_i = L_a + L_b$ (Free XOR); no material.
- $\mathsf{mux}(s, z, o)$: $L_i = H(i, L_s) + L_z$, and the material is the single ciphertext

$$T_i = H(i, L_s) + H(i, L_s + \Delta) + L_z + L_o.$$

Remark 3.5 (The multiplexer is a half-gate). Since $\mathsf{mux}(s, z, o) = z \oplus (s \wedge (z \oplus o))$, the multiplexer is an AND gate whose first input s is known to the evaluator, followed by a free XOR with z. The evaluator half-gate of Zahur, Rosulek and Evans [ZRE15, Sect. 3.1] garbles $c = a \wedge b$, with a known to the evaluator and zero-labels A, B, by choosing the zero-label $C = H(A)$ and sending the single ciphertext $H(A) \oplus H(A \oplus R) \oplus B$, where R is the global offset. With $A = L_s$, $B = L_z + L_o$ (the zero-label of $z \oplus o$) and $R = \Delta$, this is T_i, and the free XOR with z turns C into $L_i = H(i, L_s) + L_z$. The gate position i in the hash plays the role of their gate index j [ZRE15, Sect. 3.2]. Garbling every AND gate this way, with free XOR gates, is their privacy-free scheme [ZRE15, Sect. 6].

Definition 3.6 (Evaluation)**.** The evaluator knows the value and the active label of every wire. At a multiplexer with selector value v_s and active selector label S, it takes the candidate selected by v_s, with active label Y, and outputs the label $H(i, S) + Y$, plus T_i if $v_s = 1$. The active inputs for x are the labels $\mathrm{lab}_\Delta(L_j, x_j)$.

3.3 Correctness

Lemma 3.7 (One step)**.** *If every evaluated wire so far carries its plaintext value v and the label $\mathrm{lab}_\Delta(L, v)$ of its zero-label L, the same holds after evaluating one more node.*

Proof. Only the multiplexer needs a computation. If $v_s = 0$ the evaluator holds $S = L_s$ and gets $H(i, L_s) + L_z + v_z\Delta = \mathrm{lab}_\Delta(L_i, v_z)$. If $v_s = 1$ it holds $S = L_s + \Delta$ and gets

$$H(i, L_s + \Delta) + L_o + v_o\Delta + T_i = H(i, L_s) + L_z + v_o\Delta,$$

since every other term appears twice and V has characteristic 2. □

Theorem 3.8 (Correctness)**.** *For every circuit C, seed, hash H and input x, evaluating the garbling of C on the active inputs of x gives, at every node i, the plaintext value v_i and the label $\mathrm{lab}_\Delta(L_i, v_i)$.*

Proof. Induction on the nodes, with Lemma 3.7. □

Remark 3.9. No hypothesis on H or on Δ is needed: correctness is pure algebra in characteristic 2. It is the correctness of the scheme for an honest garbler.

Chapter 4

The protocol of Jawurek, Kerschbaum and Orlandi

The prover wants to convince the verifier that it knows w with $C(w) = 1$, where C is a circuit with a designated output node. We follow the protocol π_{ZK} of [JKO13] (Sect. 4, Theorem 2, and Fig. 6 of the ePrint version), which is stated in the $(\mathcal{F}_{\text{COT}}, \mathcal{F}_{\text{COM}})$-hybrid model: $\mathcal{F}_{\text{COT}}$ is an OT with an open-all phase in which the sender reveals all its inputs ([JKO13, Fig. 3], "OT with sender verifiability"), and $\mathcal{F}_{\text{COM}}$ is an ideal commitment. In π_{ZK} the prover sends its choice bits to $\mathcal{F}_{\text{COT}}$ first, and the garbled circuit is sent after the transfer. Our model below abstracts the OT and its open-all phase by the opened seed:

1. Through the OT functionality the prover receives, for each input j, the label of the bit w_j. The verifier garbles C with a seed σ and sends the material.
2. The prover evaluates and commits to the output label Z.
3. The verifier opens σ. The prover garbles again and opens Z only if the material and the OT labels are the ones that σ determines.
4. The verifier accepts if and only if Z is the 1-label $L_{\text{out}} + \Delta$ of the output node.

4.1 The model

Definition 4.1 (Messages of the verifier)**.** A possibly malicious verifier sends arbitrary material, an arbitrary pair of labels to the OT functionality for each input, and an arbitrary seed as its opening. A receiver with choice bits x gets the first or second label of each pair. The honest verifier with seed σ sends the garbling of C, the pairs $(L_j, L_j + \Delta)$, and σ.

Definition 4.2 (The prover's check and the view)**.** The honest prover evaluates the material on the labels it received. After the opening it checks that the material is the garbling of C with the opened seed and that every OT pair is $(L_j, L_j + \Delta)$. The verifier's view is the opened label if the check passes, and an abort otherwise. The simulator outputs $L_{\text{out}} + \Delta$, computed from the opened seed, if the check passes, and an abort otherwise; it does not use the witness.

In the hybrid model the OT functionality gives the verifier nothing, and the commitment functionality gives it only the fact that the prover committed. Its view is therefore the opened

label or an abort, a function of its own messages. A randomized verifier is a family of deterministic ones indexed by its coins, so equality of views for every message gives equality of view distributions.

Lemma 4.3. *If the check passes, the honest prover's label is* $\mathrm{lab}_\Delta(L_{\mathrm{out}}, v)$*, where* v *is the plaintext output on* w *and* L_{out}*,* Δ *come from the opened seed.*

Proof. OT pairs of the form $(L_j, L_j + \Delta)$ hand out the active inputs; apply Theorem 3.8. □

4.2 Completeness, zero knowledge, soundness

Theorem 4.4 (Completeness)**.** *If* $C(w) = 1$*, the honest prover passes its check and the honest verifier accepts, for every seed.*

Proof. The honest messages pass the check by definition, and Lemma 4.3 gives $Z = L_{\mathrm{out}} + \Delta$. □

Theorem 4.5 (Perfect zero knowledge against a malicious verifier)**.** *If* $C(w) = 1$*, then for every message of the verifier its view of the honest prover equals the output of the simulator. In particular the view is the same for any two witnesses.*

Proof. If the check fails both abort. If it passes, Lemma 4.3 shows that the prover's label is $\mathrm{lab}_\Delta(L_{\mathrm{out}}, 1) = L_{\mathrm{out}} + \Delta$, which is the simulator's output. □

Remark 4.6. The authenticity notion we use is the static one of [JKO13, Definition 2], which Jawurek, Kerschbaum and Orlandi describe as the authenticity of Bellare, Hoang and Rogaway [BHR12b, Sect. 3.6, game Aut of Fig. 5] restricted to their setting: the function has a one-bit output, $f(x) = 0$, and the adversary must produce the label of the output 1. In both the input x is fixed before the adversary sees the garbled circuit; the adaptive variant, where the input is chosen after, is the subject of Chapter 10. This matches π_{ZK}, where the prover's choice bits are fixed by $\mathcal{F}_{\mathrm{COT}}$ before the garbled circuit is sent.

Definition 4.7 (Cheating prover and authenticity game)**.** A cheating prover chooses its OT bits x and computes the label it commits to from the material and the labels it received. Ideal commitment makes this label independent of the later opening. In the authenticity game an adversary receives the material and the active labels of an input x of its choice, and wins if it outputs the inactive output label $\mathrm{lab}_\Delta(L_{\mathrm{out}}, 1 - v)$, where v is the plaintext output on x.

Theorem 4.8 (Soundness by reduction to authenticity)**.** *If no input satisfies* C*, then for every seed the honest verifier accepts a cheating prover if and only if the prover wins the authenticity game.*

Proof. Since the output is 0 on every input, the inactive output label is $L_{\mathrm{out}} + \Delta$, the label the verifier accepts. □

A bound on the authenticity advantage, over random seeds and a random hash and for a prover with a bounded number of hash queries, is therefore a bound on the soundness error. Chapters 5 and 6 provide one. The theorem above is for a fixed hash and an arbitrary prover; the formal composition with the bound is Theorem 7.15, which uses Theorem 6.12 directly.

4.3 Without the opening

In the warm-up of Jawurek, Kerschbaum and Orlandi [JKO13, Sect. 3] the verifier never opens its garbling: the prover sends the output label it computed, and the verifier compares it with the 1-label. This variant is zero knowledge against an honest verifier only. The two statements below make both halves of this precise.

Theorem 4.9 (Honest-verifier zero knowledge without opening)**.** *If $C(w) = 1$, then against the honest verifier with seed σ the label the prover sends is $L_{\text{out}} + \Delta$, computed from σ. It does not depend on the witness.*

Proof. The honest verifier's messages pass the prover's check, so Lemma 4.3 applies. □

Proposition 4.10 (Without the prover's check, zero knowledge fails)**.** *Let C be the circuit $x_0 \oplus x_1$, and let $a \neq 0$. A verifier that sends the honest material of C but offers the OT pairs $(0, a)$ for input 0 and $(0, 0)$ for input 1 receives the label a from a prover with witness $(1, 0)$ and the label 0 from a prover with witness $(0, 1)$. Both are witnesses, so a prover that sends its label without checking reveals its first witness bit. Whatever seed the verifier opens, the prover's check rejects these messages, since the pairs are not of the form $(L_j, L_j + \Delta)$ for a single Δ, so in the full protocol the view is an abort for both witnesses (Theorem 4.5).*

Proof. The output label is the sum of the two received labels: $a + 0$ for $(1, 0)$, and $0 + 0$ for $(0, 1)$. For an opened seed with offset Δ and input zero-labels L_0, L_1, the check would require $(0, a) = (L_0, L_0 + \Delta)$ and $(0, 0) = (L_1, L_1 + \Delta)$, hence $\Delta = 0$ and $a = \Delta = 0$. □

This is a simple attack of a malicious verifier on the input phase; the attacks that the opening is meant to prevent are listed in [JKO13, Sect. 1].

Chapter 5

Authenticity: where Δ enters the view

The evaluator can compute every active label from the material and the active inputs, and the inactive label of a wire is the active one plus Δ. Authenticity therefore amounts to the impossibility of finding Δ. In this chapter we show that Δ reaches the evaluator's view only through the hashes of the inactive selector labels.

Lemma 5.1. *The adversary wins the authenticity game if and only if it outputs $A + \Delta$, where A is the active output label that the evaluator computes from the material.*

Proof. By Theorem 3.8, $A = \mathrm{lab}_\Delta(L_{\mathrm{out}}, v)$, and $\mathrm{lab}_\Delta(L_{\mathrm{out}}, 1-v) = A + \Delta$. □

5.1 The ideal garbler

Definition 5.2 (Ideal garbler)**.** The ideal garbler runs alongside the evaluator. It receives the active input labels, the active labels of the constants, the hash H and a *pad* for each multiplexer, and it never uses Δ. At a multiplexer in position i, with active selector label S, active candidate labels Z, O and candidate values v_z, v_o, it publishes

$$T_i = H(i, S) + Z + O + \mathrm{pad}_i(S, v_z \oplus v_o).$$

Theorem 5.3 (The real material is ideal)**.** *The material of the real garbling is the material of the ideal garbler with the pads*

$$\mathrm{pad}_i(S, c) = H(i, S + \Delta) + c\,\Delta.$$

Here $S + \Delta$ is the inactive selector label of the multiplexer.

Proof. By induction on the nodes. If the two materials coincide before node i, the evaluation reaches node i with the same wires, and by Lemma 3.7 each wire carries the label $\mathrm{lab}_\Delta(L, v)$ of its zero-label L and its plaintext value v. At a multiplexer the two hashes $H(i, L_s)$, $H(i, L_s + \Delta)$ are $H(i, S)$ and $H(i, S + \Delta)$ in some order, and $Z + O + (v_z \oplus v_o)\Delta = L_z + L_o$. A case analysis on v_s, v_z, v_o shows that the entries of node i coincide too. □

Corollary 5.4. *For every seed there are pads $K_i \in V$, one per position, such that the ideal garbler with the constant pads K_i produces the real material.*

Proof. The ideal garbler reads the pad of position i only once, at position i; take K_i to be the value it reads. □

5.2 The bound in the ideal model

Theorem 5.5 (Authenticity in the ideal model)**.** *Let V be finite with $|V| \geq 2$. Fix the circuit, the active labels and H, and let $\mathcal{A}$ be any map from materials to V. For any m, let the pads of the positions $i < m$ be the entries of $K \in V^m$ and the other pads be 0. Among the pairs (Δ, K) with $\Delta \neq 0$ and $K \in V^m$, at most $|V|^m$ satisfy $\mathcal{A}(\mathsf{M}) = A(\mathsf{M}) + \Delta$, where M is the material with pads K. With Δ uniform on $V \setminus \{0\}$ and K uniform, the success probability is at most $1/(|V| - 1)$.*

Proof. The material does not depend on Δ, so for each K the condition determines Δ. □

The bound holds for every fixed value of the active labels and of H, hence on average over any distribution of them independent of (Δ, K). In the real scheme the pads are hashes at inactive points; the next chapter justifies the replacement by uniform pads in the random oracle model.

Chapter 6

Authenticity in the random oracle model

We work in the random oracle model [BR93]. The hash is now a uniformly random function $H : \{0, \dots, n-1\} \times V \to V$, where n is the number of nodes, and Δ is uniform on $V \setminus \{0\}$. The adversary sees the material, makes q adaptive queries to H, and outputs a label. The active labels of the inputs and of the constants are first fixed parameters; Theorem 6.12 at the end of the chapter removes this restriction and bounds the real garbling with uniform zero-labels.

6.1 The trace of the garbler

Definition 6.1 (Trace)**.** The trace records, for each position i that holds a multiplexer, its active selector label S_i and the XOR c_i of its candidate values. The garbler reads H at (i, S_i) and the pad at (i, S_i, c_i). In the real scheme the *inactive point* of position i is $(i, S_i + \Delta)$.

Lemma 6.2 (Congruence)**.** *Two runs of the ideal garbler whose hashes and pads agree at the points named by the trace of the first produce the same material, the same wires and the same trace.*

Proof. Induction on the nodes; a node reads only the hash and the pad of its own position. □

6.2 Adversaries and the two games

Definition 6.3 (Adversary)**.** A deterministic adversary chooses each query from the material and the previous answers, and its output from the material and all q answers. A randomized adversary is a family of these indexed by its coins.

Lemma 6.4 (Identical until bad)**.** *Two oracles that agree at every query the adversary asks give the same answers.*

This is the deterministic core of the fundamental lemma of game playing of Bellare and Rogaway [BR06, Lemma 2]; see also Shoup [Sho04]. In the proofs below it is used through injections between sets of randomness rather than through probabilities.

Proof. Induction on the number of queries. □

Definition 6.5 (Real and simulated games)**.** In the real game the material uses the pads $H(i, S_i + \Delta) + c_i \Delta$ and the adversary wins if it outputs $A + \Delta$. In the simulated game the material uses independent pads $K \in V^n$, so nothing the adversary sees depends on Δ; it wins with the same condition, and it hits if one of its queries is an inactive point $(i, S_i + \Delta)$ of the simulated trace.

6.3 The reduction

Lemma 6.6 (Moving the inactive values into the pads)**.** *Fix $\Delta \neq 0$. Given (H, J) with $J \in V^n$, let H' be H with its value at each inactive point $(i, S_i + \Delta)$ replaced by J_i, and let $K_i = H(i, S_i + \Delta) + c_i \Delta$ at multiplexers and $K_i = J_i$ elsewhere. Then the simulated game with (H', K) has the same material, wires and trace as the real game with H, and the map $(H, J) \mapsto (H', K)$ is injective.*

Proof. Active and inactive points differ because $\Delta \neq 0$, so H' agrees with H at the active points, and the pads agree by construction; Lemma 6.2 gives the first claim. Both traces coincide, so H is recovered from H' off the inactive points and from K at them, and J from H' at the inactive points and from K elsewhere. □

Lemma 6.7. *If the adversary wins the real game with H, it wins or hits in the simulated game with (H', K).*

Proof. If it does not hit, its queries avoid the inactive points, where alone H and H' differ; by Lemma 6.4 it gets the same answers, hence outputs the same label. □

Lemma 6.8. *For fixed (H', K), at most $q + 1$ values of Δ give a simulated win or a hit.*

Proof. A win fixes $\Delta = Z - A$, and the k-th query (i, y) hits only for $\Delta = y - S_i$. □

Theorem 6.9 (Authenticity in the random oracle model)**.** *Let V be finite with $|V| \geq 2$. Summed over $\Delta \neq 0$, the hash functions for which a q-query adversary outputs the inactive output label number at most $(q+1)\,|V|^{n|V|}$. With Δ uniform on $V \setminus \{0\}$ and H uniform, the success probability is at most*

$$\frac{q+1}{|V|-1}.$$

Proof. For each Δ, Lemmas 6.6 and 6.7 inject the real wins, times $|V|^n$ choices of J, into the simulated wins or hits. Summing over Δ and swapping the sums, Lemma 6.8 bounds the total by $(q + 1)$ times the number of pairs (H', K). Divide by $|V|^n$. □

6.4 The real garbling

Lemma 6.10. *The ideal garbler reads the active label of a constant only at the bit of that constant.*

Proof. Induction on the nodes. □

Lemma 6.11 (Reparametrization)**.** *Let the seed consist of Δ, zero-labels L^{in} of the inputs $j < k$ and zero-labels L^{c} of the constants; the zero-labels of the inputs $j \geq k$ are 0. Let $x_j = 0$ for $j \geq k$, so that the active labels of these inputs are 0 as well (with $x_j = 1$ the active label would be Δ itself, as for $C = [\mathrm{input}(k)]$). Then the material of the real garbling, and the game in which the adversary also receives the active labels of x, coincide with the game of Theorem 6.9 with the fixed active labels $L^{\mathrm{in}}_j + x_j \Delta$ and $L^{\mathrm{c}}_i + b_i \Delta$.*

Proof. Theorem 5.3 and Lemma 6.10. □

Theorem 6.12 (Authenticity of the real garbling)**.** *Let V be finite with $|V| \geq 2$, and let the garbler choose $\Delta \neq 0$, the hash H and the zero-labels of the inputs and of the constants uniformly. Let x be an input fixed in advance, with $x_j = 0$ for $j \geq k$ as in Lemma 6.11. An adversary that sees the material and the active labels of x, and makes q adaptive queries to H, outputs the inactive output label with probability at most $(q+1)/(|V|-1)$.*

Proof. For fixed Δ, the maps $L^{\text{in}} \mapsto L^{\text{in}} + x\Delta$ and $L^{\text{c}} \mapsto L^{\text{c}} + b\Delta$ are bijections. By Lemma 6.11 the sum over Δ and the zero-labels becomes a sum over the active labels, outside, of the sums of Theorem 6.9, inside. □

Remark 6.13. The bound of Theorem 6.9 does not use characteristic 2: it holds for labels in any finite abelian group. Theorem 6.12 does use it, through Theorem 5.3. With 128-bit labels the authenticity advantage of an adversary making q hash queries is at most $(q+1)/(2^{128}-1)$. Theorem 4.8 suggests the same bound for the soundness error of the protocol in the hybrid model; the formal statement, with the hash commitment and a prover with a bounded number of queries, is Theorem 7.15.

Chapter 7

A hash commitment instead of the ideal one

In this chapter the commitment functionality of Chapter 4 is replaced by the hash commitment $c = G(Z, r)$, where $G\colon V \times R \to D$ is a random oracle and r is uniform in a finite set R. The oblivious transfer stays ideal. Zero knowledge becomes statistical, with a simulator that programs the oracle, and soundness becomes extraction plus a binding error.

7.1 Oracle runs

Definition 7.1 (Runs)**.** A query strategy chooses each query from the previous answers. A run queries a point p if one of its queries is p.

Lemma 7.2 (Reprogramming an unqueried point)**.** *If a run does not query p, reprogramming the oracle at p changes none of its answers.*

Proof. Induction on the number of queries. □

7.2 Zero knowledge

Definition 7.3 (Malicious verifier and the three games)**.** A malicious verifier runs in three phases with adaptive queries to G: q_1 queries and then the material and the OT pairs; after the commitment, q_2 queries and then the opened seed; after the prover's last message, q_3 queries and then its output bit. In the *real* execution the prover commits with $G(Z_w, r)$, where Z_w is its output label, and sends (Z_w, r) if the opening passes the check. The *simulator* commits to a uniform c, and if the opening passes the check it computes $Z = L_{\text{out}} + \Delta$ from the seed, sends (Z, r) and programs $G(Z, r) := c$. It does not use the witness. The *hybrid* uses the real commitment point $p = (Z_w, r)$, a uniform commitment c, and the oracle $G[p := c]$ after the commitment.

Lemma 7.4. *If $C(w) = 1$, the simulator's last message is the honest prover's. In the formalization the hypothesis is that the prover's check is sound (Lemma 8.6), which for the check of Definition 4.2 follows from $C(w) = 1$; the same hypothesis is used in Lemma 7.6.*

Proof. Lemma 4.3. □

Lemma 7.5 (Real and hybrid)**.** *If the first phase does not query p, swapping the value $G(p)$ with c turns the hybrid into the real execution. The swap is an involution on these pairs (G, c), so the hybrid and the real execution output 1 equally often on them.*

Proof. By Lemma 7.2 the first phase, hence p, is the same for G and for $G[p := c]$; after the swap the real commitment is $G[p := c](p) = c$. □

Lemma 7.6 (Hybrid and simulator)**.** *The hybrid and the simulator agree unless the second phase, or the third phase after an abort, queries p.*

Proof. After a successful opening both use $G[p := c]$ in the third phase, by Lemma 7.4; elsewhere apply Lemma 7.2. □

Theorem 7.7 (Statistical zero knowledge)**.** *For every malicious verifier and every witness w with $C(w) = 1$, the probabilities that the verifier outputs 1 in the real execution and with the simulator differ by at most*

$$\frac{q_1 + q_2 + q_3}{|R|}.$$

Proof. By Lemmas 7.5 and 7.6, the two probabilities differ by at most the probability of the bad events. None of the queries involved depends on r: before the opening the simulator does not use r, and after an abort it never reveals it. So each query hits $p = (Z_w, r)$ for at most one value of r. □

7.3 Soundness

Definition 7.8 (Cheating prover and extractor)**.** A cheating prover sees a view of the garbling, makes q_1 queries to G and commits to c; after the opening it makes q_2 queries and opens a point. The verifier accepts if the opened point has the target label and its value is c. The extractor returns the label of the first first-phase query answered with c. It does not see the opening, so it is an adversary for the authenticity game.

Lemma 7.9 (Fiber bound)**.** *Let E be a set of oracles, and let $p(G)$ be a point and $f(G)$ a value that, for $G \in E$, do not change when G is reprogrammed at $p(G)$. Then $G \in E$ and $G(p(G)) = f(G)$ for at most a fraction $1/|D|$ of the oracles.*

Proof. The map $(G, x) \mapsto G[p(G) := x]$ is injective on the pairs with $G \in E$ and $G(p(G)) = f(G)$: the image determines $p(G)$, hence G off $p(G)$, and $G(p(G)) = f(G)$ is determined too. □

Lemma 7.10. *If the prover is accepted and the extractor does not win, then a fresh first-phase query got the answer of an earlier, different one, or a fresh second-phase query got the answer c, or the opened point was never queried and its value is c. Each of these events has probability at most $1/|D|$ for a fixed pair, query or point.*

Proof. If the opened point was queried in the first phase, the first query answered with c has a different label, and the first occurrence of the opened point is fresh. Otherwise the first occurrence of the opened point in the second phase, or the point itself, is fresh. In each case apply Lemma 7.9. □

Theorem 7.11 (Soundness with a hash commitment)**.** *The probability that the verifier accepts a cheating prover is at most the probability that the extractor outputs the target label, plus*

$$\frac{q_1^2 + q_2 + 1}{|D|}.$$

Proof. Sum the bounds of Lemma 7.10 over the q_1^2 pairs, the q_2 queries and the opened point. □

7.4 End-to-end soundness

Definition 7.12 (Paired prover)**.** The cheating prover receives the material and the active labels of its OT input x, and at each step makes one query to the garbling hash H and one to the commitment oracle G. A prover that interleaves its queries arbitrarily is of this form after padding. It makes n_1 steps, commits, sees the opened seed, makes n_2 more steps and opens a point. The verifier accepts if the point has the target label $L_{\text{out}} + \Delta$ and its value is the commitment.

Lemma 7.13 (Replays)**.** *For a fixed H, the prover answers its own H queries and is a prover of Definition 7.8 for the oracle G, accepted exactly when the paired prover is. For a fixed G, a strategy that answers its own G queries is an adversary for the oracle H.*

Proof. Induction on the number of steps. □

Lemma 7.14 (The extractor is an authenticity adversary)**.** *For a fixed G, the extractor, which replays the first phase and outputs the label of the first G query answered with the commitment, is an adversary of Theorem 6.12. If C is unsatisfiable and the extractor returns the target label, this adversary outputs the inactive output label.*

Proof. On an unsatisfiable circuit the active output label is L_{out}, by Theorem 3.8, so the target $L_{\text{out}} + \Delta$ is the inactive output label. □

Theorem 7.15 (End-to-end soundness)**.** *Let $|V| \geq 2$, let the seed of the verifier, the hash H and the commitment oracle G be uniform, and let C be unsatisfiable. A cheating prover with n_1 steps before the commitment and n_2 after it is accepted with probability at most*

$$\frac{n_1 + 1}{|V| - 1} + \frac{n_1^2 + n_2 + 1}{|D|}.$$

In the formalization the hypothesis on C is weaker: $C(x) = 0$ for the input x that the prover gives to the OT, which is fixed in advance, with $x_j = 0$ for $j \geq k$.

Proof. For fixed H and seed, Theorem 7.11 and Lemma 7.13 bound acceptance by the extractor's success plus $(n_1^2 + n_2 + 1)/|D|$. For fixed G, Lemma 7.14 and Theorem 6.12 bound the extractor's success by $(n_1 + 1)/(|V| - 1)$. Sum over all the randomness. □

Chapter 8

A real oblivious transfer: zero knowledge

In this chapter the ideal OT is replaced by the oblivious transfer of Chou and Orlandi [CO15, Sect. 2.1]. In their notation the sender sends $S = yB$, the receiver with choice c sends $R = cS + xB$, and the keys are $H_{(S,R)}(yR - jT)$ with $T = yS$; below $A = S$, $a = y$, $b = x$ and $g = B$. Their key hash is seeded with (S, R); ours is an arbitrary function, which is enough for zero knowledge, since the theorem holds for every key hash. Let $\mathbb{G}$ be a group with a generator g: every element is $b \cdot g$ for a unique $b \in \mathbb{Z}_q$. In this chapter q need not be prime; Chapter 11 assumes it. For each input j the verifier, as sender, sends $A = a \cdot g$; the prover, as receiver with choice bit w, sends $B = b \cdot g + wA$; the verifier sends the two labels masked with the keys $H_K(a \cdot B)$ and $H_K(a \cdot (B - A))$, and the prover unmasks the label of w with $H_K(b \cdot A)$. When the verifier opens its seed it also opens its OT secrets a, so the prover checks both masked labels, not only its own: a verifier cannot make the prover abort depending on w.

8.1 The oblivious transfer

Definition 8.1 (Chou–Orlandi OT)**.** The receiver's message is $B = b \cdot g + wA$, and it unmasks $e_w - H_K(b \cdot A)$. For the analysis we write $m_c(A, B) = e_c - H_K(\log_g(B - cA) \cdot A)$, the label of the choice c unmasked with the key of a receiver whose message is B.

Lemma 8.2 (Receiver privacy)**.** *For every A and w, the map $b \mapsto b \cdot g + wA$ is a bijection. In particular B is uniform whatever w is. This is [CO15, Lemma 1].*

Proof. A bijection followed by a translation. □

Lemma 8.3. *The receiver obtains $m_w(A, B)$, a function of the sender's messages and of B only. If $A = a \cdot g$, then $m_c(A, B) = e_c - H_K(a \cdot (B - cA))$, which the receiver can compute once a is opened.*

Proof. $\log_g(B - wA) = b$, and $\log_g(X) \cdot (a \cdot g) = a \cdot X$. □

8.2 Zero knowledge

Lemma 8.4 (Locality of evaluation)**.** *If every input node of C reads an input $j < k$, the evaluation depends only on the input wires $j < k$.*

Proof. Induction on the nodes. □

Definition 8.5 (Verifiers with the real OT)**.** A malicious verifier makes q_0 queries to G and sends the messages A; after the receiver messages B it makes q_1 queries and sends the material and the masked pairs; the remaining phases are those of Definition 7.3, and the opening also contains the OT secrets. For fixed B it is a verifier of Chapter 7 whose OT pairs are $(m_0(A,B), m_1(A,B))$. The prover's check verifies $A = a \cdot g$, that the material is the garbling of the opened seed, and that both masked labels of every input unmask, with the sender's keys, to $(L_j, L_j + \Delta)$. It does not use the witness.

Lemma 8.6 (The check is sound)**.** *If C reads only the inputs $j < k$ and $C(w) = 1$, then whenever the check passes the honest prover's label is the 1-label $L_{\text{out}} + \Delta$ of the opened seed.*

Proof. By Lemma 8.3 the prover received the active labels of w on the inputs $j < k$; apply Lemma 8.4 and Theorem 3.8. □

Theorem 8.7 (Zero knowledge with the real OT and the hash commitment)**.** *For every malicious verifier and every witness w of a circuit that reads only the inputs $j < k$, the probabilities that the verifier outputs 1 in the real execution and with the simulator differ by at most*

$$\frac{q_0 + q_1 + q_2 + q_3}{|R|}.$$

The simulator chooses the receiver messages B uniformly and runs the simulator of Theorem 7.7.

Proof. The proof of Theorem 7.7 uses only that the prover's check is sound; with Lemma 8.6 it applies for each fixed B. By Lemma 8.2, summing the real execution over the receiver secrets b is summing over uniform B. □

Remark 8.8. Receiver privacy is perfect, and zero knowledge holds for every key hash H_K.

8.3 Sender security: the square Diffie–Hellman reduction

Lemma 8.9. *The two key points of the sender differ by the square Diffie–Hellman value: $a \cdot B - a \cdot (B - a \cdot g) = (aa) \cdot g$. This is the observation behind the proof of [CO15, Lemma 2].*

Proof. Linearity of the scalar multiplication. □

Theorem 8.10 (Learning both labels solves square Diffie–Hellman)**.** *Let a receiver make n queries to the key hash, at points $Q_0, \dots, Q_{n-1}$ that, like its message B, are functions of A alone: it does not adapt to the answers of the hash or to the masked labels (the general case is Chapter 11). For uniform a, the probability that it queries both key points of an input is at most the sum, over the n^2 pairs of query positions (i, j), of the probability that the solver $A \mapsto Q_i - Q_j$ computes $(aa) \cdot g$ from $a \cdot g$; hence at most n^2 times the best square Diffie–Hellman advantage.*

Proof. If $Q_i = a \cdot B$ and $Q_j = a \cdot (B - A)$, Lemma 8.9 gives $Q_i - Q_j = (aa) \cdot g$. □

Remark 8.11 (Square and computational Diffie–Hellman). In a group of odd prime order q the two problems are equivalent; Bao, Deng and Zhu give randomized reductions in both directions [BDZ03, Sect. 2.1]. The elementary argument below is not formalized here, and it applies as written only to solvers that always succeed. A computational Diffie–Hellman solver, applied to (ag, ag), gives $a^2 g$. Conversely, from a square solver S, $ab \cdot g = 2^{-1} \cdot (S((a+b)g) - S(ag) - S(bg))$, since $(a+b)^2 - a^2 - b^2 = 2ab$. For solvers that succeed only with some probability the three calls must be rerandomized, as in [BDZ03, Sect. 2.1].

Remark 8.12. This is the first of three steps towards soundness with the real OT. The other two are in Chapters 10 and 11: the authenticity bound of Chapter 6 is extended to inputs that the prover chooses adaptively, when it queries the key points after seeing the material, and in the random oracle model a label whose key point is never queried is hidden by a uniform mask.

Chapter 9

A real oblivious transfer: the dual-mode OT of Peikert, Vaikuntanathan and Waters

In the protocol of [JKO13] the prover's choice bits are fixed by $\mathcal{F}_{\mathrm{COT}}$ before the garbled circuit is sent, and this is what makes static authenticity enough (Chapter 4). Jawurek, Kerschbaum and Orlandi describe how to realize $\mathcal{F}_{\mathrm{COT}}$ with the OT of Peikert, Vaikuntanathan and Waters [PVW08], adding an open-all phase in which the sender reveals its coins [JKO13, Appendix A]. In this chapter we take its DDH instance [PVW08, Sect. 5.3] in messy mode, where the receiver's choice can be extracted from its first message with a trapdoor and the sender's security is statistical [PVW08, Theorem 4.1]. We show that soundness then reduces exactly, with no computational step, to the end-to-end bound of Theorem 7.15. For zero knowledge, the receiver's security is computational in messy mode; we reduce it to the indistinguishability of the two modes of the common string, which [PVW08, Theorem 5.2] derives from DDH.

9.1 The dual-mode cryptosystem

We write the group additively: $\mathbb{G}$ has prime order q and is a $\mathbb{Z}_q$-module.

Definition 9.1 (Messy common string, keys, encryption)**.** A common string is a tuple (g_0, h_0, g_1, h_1) of group elements. It is messy, with trapdoor (x_0, x_1), if $g_0, g_1 \neq 0$, $h_b = x_b \cdot g_b$ and $x_0 \neq x_1$; only the extraction uses the trapdoor. The receiver with choice σ and secret r sends $pk = (r \cdot g_\sigma, r \cdot h_\sigma)$. The encryption of $m \in \mathbb{G}$ on branch b with coins (s, t) is

$$(s \cdot g_b + t \cdot h_b,\ s \cdot pk_1 + t \cdot pk_2 + m),$$

and (c_0, c_1) decrypts with r to $c_1 - r \cdot c_0$. This is [PVW08, Sect. 5.3], where encryption is built on the algorithm Randomize of [PVW08, Lemma 5.1].

Lemma 9.2 (Correctness)**.** *Decryption is correct on the receiver's branch.*

Proof. $s \cdot (r \cdot g_\sigma) + t \cdot (r \cdot h_\sigma) = r \cdot (s \cdot g_\sigma + t \cdot h_\sigma)$. □

Lemma 9.3 (Messy branches)**.** *If $g_b \neq 0$ and $pk_2 \neq x_b \cdot pk_1$, then for every message m the encryption on branch b is a bijection from the coins $\mathbb{Z}_q^2$ onto $\mathbb{G}^2$. In particular, with uniform*

coins the ciphertext is uniform whatever m is. This is the second case of [PVW08, Lemma 5.1], in exact form.

Proof. Both sides have q^2 elements, so it suffices to show injectivity. Encryption is affine in the coins. If the difference (δ_s, δ_t) of two coins is mapped to 0, then $(\delta_s + \delta_t x_b) \cdot g_b = 0$, so $\delta_s = -\delta_t x_b$, and then $\delta_t \cdot (pk_2 - x_b \cdot pk_1) = 0$, so $\delta_t = 0$. □

Lemma 9.4 (Extraction)**.** *Let* $\mathrm{ext}(pk)$ *be* 0 *if* $pk_2 = x_0 \cdot pk_1$ *and* 1 *otherwise. If the common string is messy and* $pk_1 \neq 0$*, the branch* $1 - \mathrm{ext}(pk)$ *satisfies the hypothesis of Lemma 9.3. On the other hand, for* $pk = (0, 0)$ *the second component of every ciphertext is the message.*

Proof. If $pk_2 = x_0 \cdot pk_1 = x_1 \cdot pk_1$, then $(x_0 - x_1) \cdot pk_1 = 0$, which is impossible. The second claim is immediate. □

Remark 9.5. ext is the complement of the algorithm FindMessy of [PVW08, Sect. 5.3], which outputs the messy branch $1 - \mathrm{ext}(pk)$. The argument that $pk_2 = x_0 \cdot pk_1$ implies $pk_2 \neq x_1 \cdot pk_1$ needs $pk_1 \neq 0$, and the key $(0, 0)$ (the pair $(1, 1)$ in multiplicative notation) has no messy branch. The version of August 2008 does not check this. The revision of January 2019 does: its sender outputs nothing when pk_1 is the identity, its FindMessy treats that case apart, and its receiver chooses $r \neq 0$ [PVW08, Sect. 5.3]. We follow the revision on the sender's side: the verifier rejects any key with $pk_1 = 0$. On the receiver's side the honest prover of Definition 9.13 samples r in all of $\mathbb{Z}_q$, as in [JKO13, App. A], so with probability at most k/q it sends a key that the verifier rejects; no completeness statement is made for this OT. Lemma 9.2 holds for every r.

Definition 9.6 (Decryption mode)**.** A common string is in decryption mode if $h_b = x \cdot g_b$ for a common x and $g_1 = y \cdot g_0$ with $y \neq 0$. Then the key of the choice 1 with secret r is the key of the choice 0 with secret yr (trapdoor key generation, [PVW08, Sect. 5.3]).

9.2 Soundness

Definition 9.7 (The protocol with the PVW OT)**.** The common string is messy and fixed. The cheating prover sends keys pk_j, one for each input; the verifier rejects if some $(pk_j)_1 = 0$. Otherwise it sends, for each input j and each branch b, the encryption on branch b of $\iota(L_j + b\Delta)$, where ι is any map from labels to $\mathbb{G}$, with uniform coins. The rest is the protocol of Theorem 7.15, except that after the opening the prover also sees all the coins of the OT, as in the open-all phase of [JKO13, Appendix A].

Lemma 9.8 (Sampling the messy ciphertext)**.** *Fix the labels. Sampling the coins of both branches is the same as sampling the ciphertexts of the messy branches and the coins of the other branches: the coins of a messy branch are the preimage of its ciphertext.*

Proof. Lemma 9.3, input by input. □

Theorem 9.9 (Soundness with the PVW OT in messy mode)**.** *For an unsatisfiable circuit and a messy common string, with the seed, the hashes and the coins of the OT uniform, the verifier accepts with probability at most*

$$\frac{n_1 + 1}{|V| - 1} + \frac{n_1^2 + n_2 + 1}{|D|},$$

the bound of Theorem 7.15.

Proof. Fix the sampled messy ciphertexts and the coins of the other branches. The prover is then a prover of Theorem 7.15 with OT input $\mathrm{ext}(pk_j)$: before the opening it computes the ciphertext of the extracted branch from the active label, and after the opening it computes the coins of the messy branch from the seed (Lemma 9.8). Apply Theorem 7.15 and average. □

Remark 9.10. The reduction is not efficient: it inverts encryption on the messy branch, which amounts to computing discrete logarithms. This does not matter here, since the bound of Theorem 7.15 holds for every prover with the given numbers of queries. Lemma 9.2 is not used in the proof; it is the completeness of the OT.

9.3 Zero knowledge

Definition 9.11 (Verifiers with the PVW OT)**.** A malicious verifier sees the prover's keys $pk_j = \mathrm{keyGen}(w_j, r_j)$; the rest is as in Definition 7.3, with the ciphertexts of both branches in its first message and all the coins of the OT in its opening. The prover decrypts the label of its choice and decodes it with a left inverse of ι. Its check after the opening: the material is the garbling of the opened seed, and every ciphertext is the encryption of the label of its branch with the opened coins. It does not use the witness.

Lemma 9.12 (The check is sound)**.** *If C reads only the inputs $j < k$ and $C(w) = 1$, then whenever the check passes the honest prover's label is the* 1*-label of the opened seed.*

Proof. The ciphertext of the prover's branch is an encryption under its own key, so it decrypts to the active label (Lemma 9.2); apply Lemma 8.4 and Theorem 3.8. □

Since the check recomputes every ciphertext from the seed and the coins, the opening cannot be equivocated, and no binding argument for the open-all phase [JKO13, Appendix A] is needed.

Definition 9.13 (Hybrids)**.** For choice bits x, the hybrid Hyb_x sends the keys of x and runs the simulator of Theorem 7.7. The simulator of the protocol is Hyb_0: it sends the keys of the choice 0 and does not use the witness. As a function of the common string, the probability p_x that the verifier outputs 1 in Hyb_x is computed without the witness (beyond the bits x) and without any trapdoor.

Theorem 9.14 (Zero knowledge for a fixed common string)**.** *For every common string, every malicious verifier and every witness of a circuit that reads only the inputs $j < k$,*

$$|p_{\mathrm{real}} - p_0| \le \frac{q_1 + q_2 + q_3}{|R|} + |p_w - p_0|.$$

Proof. For fixed keys the verifier is a verifier of Chapter 7, and Lemma 9.12 makes Theorem 7.7 apply: the real execution and Hyb_w differ by at most $(q_1 + q_2 + q_3)/|R|$. Then use the triangle inequality. □

Lemma 9.15 (Decryption mode)**.** *In decryption mode p_x does not depend on x.*

Proof. By Definition 9.6, the map $r_j \mapsto yr_j$ on the inputs with $x_j = 1$ is a bijection of the secrets that turns the keys of x into the keys of 0, and the simulator does not use the secrets otherwise. □

Theorem 9.16 (Zero knowledge in messy mode, by switching modes)**.** *Let the verifier depend on the common string and make at most Q queries. Averaged over a nonempty set S_m of common strings, the real execution and the simulator differ by at most*

$$\frac{Q}{|R|} + |\mathbb{E}_{S_m} p_w - \mathbb{E}_{S_d} p_w| + |\mathbb{E}_{S_d} p_0 - \mathbb{E}_{S_m} p_0|$$

for every set S_d of decryption-mode strings.

Proof. The first step of the proof of Theorem 9.14 gives $|p_{\text{real}} - p_w| \le Q/|R|$ for each common string; averaging over S_m, $|\mathbb{E}_{S_m} p_{\text{real}} - \mathbb{E}_{S_m} p_w| \le Q/|R|$. Then use the triangle inequality through $\mathbb{E}_{S_d} p_w = \mathbb{E}_{S_d} p_0$ (Lemma 9.15). Averaging the bound of Theorem 9.14 itself would leave the term $\mathbb{E}_{S_m} |p_w - p_0|$, which the two differences of averages do not control. □

Remark 9.17. With S_m and S_d the supports of the messy and the decryption-mode setups, the last two terms are the advantages of the distinguishers p_w and p_0 between the two modes. The next section bounds them by DDH advantages.

9.4 The two modes and DDH

We follow [PVW08, Sect. 5.1 and 5.3], in the 2019 revision, in a group of prime order $q > 2$. The messy setup samples $g_0, g_1 \neq 0$ and distinct nonzero x_0, x_1; the decryption setup samples $g_0 \neq 0$ and nonzero x, y, and outputs (g_0, xg_0, yg_0, xyg_0). The DDH assumption of [PVW08, Sect. 5.1] says that for random $g, h \neq 0$ the tuples (g, h, ag, ah) with a random and (g, h, ag, bh) with $a \neq b$ random are computationally indistinguishable. For a function F of the tuple with values in $[0, 1]$ we write $\text{Adv}_{\text{DDH}}(F)$ for the absolute value of the difference of the averages of F over the tuples of the first kind and over those of the second.

Lemma 9.18 (The setups as tuples)**.** *The messy setup is the distribution of random tuples conditioned on $a, b \neq 0$, and the decryption setup is the distribution of DDH tuples conditioned on $a \neq 0$.*

Proof. With $h = x_0 g_0$, $g_1 = a g_0$ and $b = a x_1 / x_0$, the map $(g_0, g_1, x_0, x_1) \mapsto (g_0, h, a, b)$ is a bijection onto the tuples with $g, h \neq 0$ and $a \neq b$ nonzero; similarly $(g_0, y, x) \mapsto (g_0, x g_0, y)$. This is the computation in the proof of [PVW08, Theorem 5.2], where it is stated as statistical closeness. □

Theorem 9.19 (Indistinguishability of the modes)**.** *For every function F of the common string with values in $[0, 1]$, the averages of F over the messy and the decryption setups differ by at most the DDH advantage of F, read as a function of the tuple, plus $3/q$.*

Proof. By Lemma 9.18, conditioning removes a proportion $2/q$ of the random tuples and $1/q$ of the DDH tuples, and the average of a function with values in $[0, 1]$ over a subset moves by at most the proportion removed. □

Theorem 9.20 (Zero knowledge in messy mode under DDH)**.** *Let the verifier depend on the common string and make at most Q queries. Averaged over the messy setup, the real execution and the simulator differ by at most*

$$\frac{Q}{|R|} + \text{Adv}_{\text{DDH}}(p_w) + \text{Adv}_{\text{DDH}}(p_0) + \frac{6}{q}.$$

Proof. As in Theorem 9.16, with Theorem 9.19 applied to p_w and p_0. □

Remark 9.21. The distinguishers p_w and p_0 are efficient when the verifier is: they sample keys and run the verifier with the simulator, and use no trapdoor. Lean has no notion of efficient computation, so this is not checked; under DDH their advantages are negligible.

Chapter 10

Adaptive authenticity

Status. No originality is claimed for this chapter. We expect a bound of this kind to be known to specialists, but we have not found a published statement of it (see below). Everything in it is formalized: the reduction to the bad event (Section 10.3) and the bound on the bad event (Section 10.4).

Where the problem comes from. In the protocol of [JKO13] the problem does not arise: the OT is the functionality $\mathcal{F}_{\mathrm{COT}}$, which fixes the prover's choice bits before the garbled circuit is sent ([JKO13, Fig. 3 and Fig. 6]), and the authenticity needed is static [JKO13, Definition 2]. It arises only when $\mathcal{F}_{\mathrm{COT}}$ is replaced by an OT from which the receiver's choice cannot be extracted when its message is sent, as with the Chou–Orlandi protocol of Chapter 8, whose receiver message B does not fix the choice. As Byali, Patra, Ravi and Sarkar [BPRS17, Sect. 3] and Doerner, Kondi, Lee and shelat [DKLs18, App. A] point out, a corrupted receiver can delay the decryption, so a simulator cannot extract its choice before the end of the protocol. Chou and Orlandi acknowledge the issue in the revised version of their paper [CO15, Sect. 1.1], where the attacking party is called the sender, and note that it matters, for instance, when the OT is combined with garbled circuits. They withdraw the claim of UC security and suggest, as a remedy, a receiver message that proves timely decryption. This is a different issue from the one raised by Genç, Iovino and Rial [GIR], who point out a flaw in the UC security proof of [CO15] for a corrupted *sender*; a claim of theirs about a corrupted receiver, made in an earlier version, was withdrawn [GIR, Sect. 1]. An OT that realizes $\mathcal{F}_{\mathrm{COT}}$ avoids it; Chapter 9 does this with the OT of [PVW08].

With the Chou–Orlandi OT the receiver's message B does not commit to a choice: the prover receives both masked labels of every input together with the material, and it decides which key point of H_K to query, that is, which label to unmask, after seeing the material. Soundness with the real OT therefore needs authenticity when the input x is chosen *adaptively*, after the material and possibly after queries to H. The bound of Chapter 6 is for a fixed x, and its proof writes the material through the ideal garbler for that x (Theorem 5.3), which is circular when x is chosen later. Bellare, Hoang and Rogaway [BHR12] define adaptive authenticity in two forms [BHR12, Sect. 4, Fig. 6]: aut1, where the adversary chooses x after seeing the garbled circuit, and aut2, where the bits of x are chosen one at a time and each one is answered with its token. The game below is aut2, with access to the random oracle added. Their results (Theorems 5 and 6, proved in their full version) are transforms that turn any statically secure scheme into an adaptively secure one, by masking the garbled circuit until the input is given; they do not say whether privacy-free Free-XOR garbling itself is aut2-secure, which is what we would need here. More recently, Barnum, Heath, Kolesnikov and Ostrovsky prove adaptive security of unmodified

schemes, half-gates among them, in the non-programmable random oracle model [BHKO26]; but their notion is adaptive *privacy* [BHKO26, Definition 5] (and obliviousness), not authenticity, and their argument ([BHKO26, Theorem 9]: query hiding plus private rekeying) is of the same kind as the bad event of Section 10.4. Guo, Yang, Wang, Yu and Liu prove that unmodified half-gates (the version with privacy, with a hash built from a random permutation) is adaptively secure in the non-programmable random permutation model [GYWYL, Sect. 5, Theorem 1]; their notion is again simulation-based privacy [GYWYL, Definition 1], and their method, the H-coefficient technique, bounds the statistical distance between the real and the simulated transcripts. We have not found a published statement of adaptive authenticity for privacy-free garbling.

10.1 The game

Definition 10.1 (Adaptive authenticity game)**.** The garbler samples $\Delta \in V \setminus \{0\}$, the zero-labels of the inputs and of the constants, and a random oracle H, and sends the material M. The adversary then interleaves, in any order chosen adaptively:

- queries (i, y) to H;
- label requests: for an input j not yet requested and a bit b of its choice, it receives the label $L_j + b\Delta$.

It takes q actions in all, queries and requests together. In the formalization a repeated request of an input is answered with the label given the first time, whatever the bit. At the end the requested bits form x (unrequested bits are 0) and the adversary outputs a label Z. It wins if $Z = X_{\text{out}} + \Delta$, where X_{out} is the active output label for x.

If the adversary has requested every input that the circuit reads, it can compute X_{out} from the material, the requested labels and the queries (i, S_i) at the active selector labels, and then winning amounts to finding Δ. In general it must produce $X_{\text{out}} + \Delta$, and an unrequested input read by the circuit only makes this harder.

Remark 10.2 (The model). The hash is a uniformly random function H and the adversary is computationally unbounded; only its number q of actions is bounded. Authenticity is a game, not a simulation-based notion: no simulator answers the adversary's queries, so the oracle is never programmed. The simulated game of Definition 10.3 and the maps between randomness of Lemmas 10.5 and 10.15 are devices of the proof, which counts the values of (Δ, L, L_c, H) for which the adversary wins; the bound is a statement about the real game with H uniform. The distinction between programmable and non-programmable random oracles, which matters for the simulation-based privacy results of [BHKO26] and [GYWYL], does not arise here. What the bound does assume is the random oracle model itself, with H used only by the garbling scheme; instantiations of H are not covered (see Chapter 14), and in a protocol that uses a hash elsewhere the garbling must hash in a separate domain (Remark 10.18). Circuits are lists of nodes in evaluation order, and every operand refers to an earlier position; "position" and "largest i" below refer to this order.

10.2 The simulated game

Definition 10.3 (Simulated game)**.** The simulator sends uniform ciphertexts T_i and uniform active constant labels, answers every label request with a fresh uniform label R_j whatever the bit, and answers queries with a random function H'. It never uses Δ; we sample Δ at the end.

All active labels are defined by evaluation: for an input x, the active label of an input j is R_j, and the evaluator reads H' at the active selector labels.

10.3 The real game without bad queries

The zero-labels L_s of the simulated game of Definition 10.3 depend on the final choice x, so two runs with different completions do not share them. This breaks a direct extension of the injection of Chapter 6: the map from real to simulated randomness has to know x to move the hash values at the inactive points, and on a run with a bad query x cannot be read back from the image. We therefore take the bad event in the *real* game, where the two points L_s, $L_s + \Delta$ of every selector are fixed by the garbler's randomness, and for the final x.

Definition 10.4 (The adaptive game in Lean)**.** The adversary is deterministic (a randomized one is a family indexed by its coins) and takes q actions, each a query (i, y) to H or a request (j, b) for an input $j < m$. Only the first request of an input counts: later requests of the same input are answered with the label of the first bit. The real randomness is (L, L_c, H, J): the zero-labels of the inputs and of the constants, the hash and a junk component, which the real game does not use. A query (i, y) is *bad for the final input* if $y = S_i + \Delta$, where S_i is the active label, for the final x, of the selector of the multiplexer at position i. The simulated game is the one of Definition 10.3, with randomness (R, A, H', T): requested labels, active constant labels, hash and ciphertexts.

Section 10.4 bounds the probability of a query bad for the final input.

Lemma 10.5 (Identical until bad)**.** *For $\Delta \neq 0$, map (L, L_c, H, J) to (R, A, H', T), where $R_j = L_j + x_j\Delta$ for the final x, A are the active constant labels, H' is H with its values at the inactive points of the final x replaced by J (Lemma 6.6), and T_i is the real ciphertext of node i if it is a multiplexer and the pad J_i of Lemma 6.6 otherwise; the injectivity below uses both. The simulated material of the image is the real material. Without a bad query, the simulated game on the image runs exactly as the real game, and a real win is a simulated win.*

Proof. Requests are answered with R_j in both games, since the first requested bit is the final one. Queries are answered by H and H', which differ only at the inactive points. The active output label is the same with H and with H', because the evaluator reads the hash only at active points. □

Lemma 10.6 (Injectivity)**.** *For $\Delta \neq 0$ the map of Lemma 10.5 is injective on the real randomness without a bad query.*

Proof. The image determines the run of the simulated game, which is the real run, hence the final x. Then R and A give L and L_c. The material determines the pads of Lemma 6.6 at the multiplexers, and T gives them elsewhere; with H', injectivity of the map of Lemma 6.6 gives (H, J). □

Theorem 10.7 (Winning without a bad query)**.** *Let V be finite. With Δ uniform on $V \setminus \{0\}$ and the zero-labels and the hash uniform, the adversary wins without a bad query with probability at most $1/(|V| - 1)$. Hence it wins with probability at most $1/(|V| - 1)$ plus the probability of a bad query.*

Proof. For each $\Delta \neq 0$, Lemmas 10.5 and 10.6 inject the real wins without a bad query into the simulated wins. The simulated game does not use Δ, so each simulated randomness wins for at most one Δ. Sum over Δ and swap the sums. □

With Theorem 10.7, Theorem 10.17 reduces to Theorem 10.16, the bound on bad queries.

10.4 Bad queries

We bound the probability of a query that is bad for the final input. The difficulty is that this event is not decided at the time of the query: whether a point is inactive depends on bits the adversary requests later. We replace it by an event that is decided at the time of the query, and for that we need to know which selector labels the adversary can know at that time. A selector whose *value* is fixed by the bits requested so far does not qualify: the XOR of two wires that always carry opposite values has a fixed value, but its label may involve hash values the adversary has never seen. The right notion is a combinatorial "linear part" of each label.

Fix the adversary and a step k. Let R_k be the set of inputs requested before step k, and $x^{(k)}$ the input formed by the bits requested before step k, with 0 elsewhere. For a wire w, $v_w(x)$ is its value on the input x and $X_w^{(k)}$ its active label for $x^{(k)}$; in the real game $X_w^{(k)} = Z_w + v_w(x^{(k)})\Delta$, where Z_w is the zero-label. For a multiplexer at position i, s_i is its selector and c_i the candidate selected by $v_{s_i}(x^{(k)})$.

Definition 10.8 (Linear part)**.** The *atoms* are the symbols a_j, for the inputs j, and b_i, for the multiplexers i. The linear part $\ell_w^{(k)} \in \mathbb{F}_2^{\text{atoms}}$ of a wire is defined by induction:

- an input j: a_j if $j \notin R_k$, and 0 if $j \in R_k$; a constant: 0;
- an XOR $w = a \oplus b$: $\ell_a^{(k)} + \ell_b^{(k)}$;
- a multiplexer i: $\ell_{c_i}^{(k)}$ if $\ell_{s_i}^{(k)} = 0$, and $b_i + \ell_{c_i}^{(k)}$ otherwise.

A wire is *known* at step k if $\ell_w^{(k)} = 0$, and *unknown* otherwise. The linear part depends only on the circuit, R_k and $x^{(k)}$, not on the randomness.

The atom b_i stands for the hash value $H(i, X_{s_i}^{(k)})$ read by the evaluator at an unknown selector, and a_j for the label of an input not yet requested. Lemma 10.14 makes this precise.

Lemma 10.9 (Known wires are determined)**.** *Let x' agree with $x^{(k)}$ on R_k. There is $\varphi : \text{atoms} \to \mathbb{F}_2$ with $v_w(x') + v_w(x^{(k)}) = \sum_\beta \ell_w^{(k)}(\beta)\varphi(\beta)$ for every wire w. In particular a known wire has the same value, and in the real game the same active label, on x' and on $x^{(k)}$.*

Proof. Put $\varphi(a_j) = x'_j + x_j^{(k)}$ and $\varphi(b_i) = v_i(x') + v_i(x^{(k)}) + v_{c_i}(x') + v_{c_i}(x^{(k)})$, and argue by induction on the position. Inputs, constants and XORs are immediate. At a multiplexer with an unknown selector the claim is the definition of $\varphi(b_i)$ together with the induction hypothesis for c_i. At a multiplexer with a known selector the selector has the same value on x' and on $x^{(k)}$ by induction, so both select c_i, and $v_i = v_{c_i}$ on both inputs. The active label is $Z_w + v_w\Delta$. □

Lemma 10.10 (Monotonicity)**.** *For $j \le k$ there is a linear map ψ with $\ell_w^{(k)} = \psi(\ell_w^{(j)})$ for every wire w. In particular a wire known at step j is known at step k.*

Proof. Put $\psi(a_t) = a_t$ if $t \notin R_k$ and 0 otherwise, and $\psi(b_i) = \ell_i^{(k)} + \ell_{c_i^{(j)}}^{(k)}$, where $c_i^{(j)}$ is the candidate selected at step j. By induction on the position: XORs are linear; at a multiplexer with a selector unknown at step j, $\psi(\ell_i^{(j)}) = \psi(b_i) + \psi(\ell_{c_i^{(j)}}^{(j)}) = \ell_i^{(k)} + \ell_{c_i^{(j)}}^{(k)} + \ell_{c_i^{(j)}}^{(k)} = \ell_i^{(k)}$. At a multiplexer with a selector known at step j, the selector is known at step k (its linear part is $\psi(0) = 0$), and by Lemma 10.9, since $x^{(k)}$ agrees with $x^{(j)}$ on R_j, it has the same value at both steps; so the same candidate c is selected, and $\ell_i^{(k)} = \ell_c^{(k)} = \psi(\ell_c^{(j)}) = \psi(\ell_i^{(j)})$. □

Definition 10.11 (Bad step)**.** Step k of the real game is *bad* if its action is a query (i, y) at a multiplexer i with selector s, and either s is known at step k and $y = X_s^{(k)} + \Delta$, or s is unknown at step k and $y \in \{Z_s, Z_s + \Delta\} = \{X_s^{(k)}, X_s^{(k)} + \Delta\}$.

Lemma 10.12 (First bad step)**.** *If some query is bad for the final input, some step is bad. If k is the first bad step, the game truncated to its first k steps has no query bad for its final input, which is $x^{(k)}$.*

Proof. Let the query at step k be bad for the final input x: $y = X_s + \Delta$ for the active label X_s of s on x. Since x agrees with $x^{(k)}$ on R_k, if s is known at step k then $X_s = X_s^{(k)}$ by Lemma 10.9; otherwise y is one of $Z_s, Z_s + \Delta$ anyway. For the second claim, let $j < k$ and let the query at step j be (i, y) with selector s. If s is known at step j, then $X_s^{(k)} = X_s^{(j)}$ by Lemma 10.9, and $y \neq X_s^{(j)} + \Delta$ because step j is not bad. If s is unknown at step j, y is neither of its two points. □

Lemma 10.13 (The image of a first bad step)**.** *Let $\Delta \neq 0$ and $k < q$. The map of Lemma 10.5 for the game truncated to k steps is injective on the real randomness whose first bad step is k. On its image the simulated game, for the same k steps and the same action at step k, has: the action at step k is a query (i, y) at a multiplexer, and, with X the simulated active label of its selector s_i for $x^{(k)}$, either $y = X + \Delta$, or s_i is unknown at step k, $y = X$, and no query before step k is at a point $(i', X_{s_{i'}})$ of a multiplexer i' whose selector is unknown at step k.*

Proof. Injectivity is Lemma 10.6 for the truncated game, which has no bad query by Lemma 10.12. The simulated run on the image coincides with the real run for k steps (Lemma 10.5), so the action at step k is the same, and the active labels for $x^{(k)}$ are the same in both games, because the evaluator reads the hash only at active points. A selector unknown at step k is unknown at every earlier step (Lemma 10.10), so a query at one of its points before step k would be a bad step. □

Lemma 10.14 (Shifting an atom)**.** *In the simulated game at step k, give the atoms the values $\alpha(a_j) = R_j$ and $\alpha(b_i) = H'(i, X_{s_i})$, the labels being those for $x^{(k)}$. Let s be an unknown wire, and let β^* be b_{i^*} for the largest i^* with b_{i^*} in $\ell_s^{(k)}$ if there is one, and some a_j in $\ell_s^{(k)}$ otherwise. Add v to the randomness at β^*: to R_j, or to H' at the point $(i^*, X_{s_{i^*}})$. Then the labels of the wires before position i^* do not change, and X_s changes by exactly v.*

Proof. Write δ for the change. By induction on the position, $\delta X_w = \sum_\beta \ell_w^{(k)}(\beta)\, \delta\alpha(\beta)$. XORs are linear. At a multiplexer with an unknown selector, $X_i = \alpha(b_i) + X_{c_i}$, plus T_i if the selector is 1. At a multiplexer with a known selector, $X_i = H'(i, X_{s_i}) + X_{c_i}$ (plus T_i); the selector label does not change (its linear part is 0), and neither does H' at gate i, since $i \neq i^*$. The atoms change as follows: $\delta\alpha(\beta^*) = v$; the other a_j do not change; $\delta\alpha(b_i) = 0$ for $i < i^*$, because the labels before i^* do not change (by the same induction, the linear parts of those wires involve only such atoms). Since $\ell_s^{(k)}$ contains no b_i with $i > i^*$, $\delta X_s = v$. □

Lemma 10.15 (Fresh labels)**.** *For each k, the simulated randomness for which the action at step k is a query (i, y) with s_i unknown at step k, $y = X_{s_i}$, and no earlier query at the active label of a selector unknown at step k (the condition of Lemma 10.13; in particular none at the point of β^* for $s = s_i$, when β^* is some b_{i^*}), is at most a fraction $1/|V|$ of all of it.*

Proof. Map a pair (σ, v), with σ in this set and $v \in V$, to σ shifted by v at β^*. The shift does not change the view up to step k: R_j is not requested before step k, and H' changes at a point

not queried before step k. So the image determines the run up to step k, hence $x^{(k)}$, the linear parts, β^* and its point (the labels before i^* do not change). Then $v = X_{s_i} - y$ on the image, by Lemma 10.14, and σ is recovered. So the map is injective, and the set has at most $|\mathrm{Sim}|/|V|$ elements. □

Theorem 10.16 (Bad queries)**.** *With Δ uniform on $V \setminus \{0\}$ and the rest uniform, a query bad for the final input occurs with probability at most $q(1/(|V|-1)+1/|V|)$, for an adversary taking q actions.*

Proof. By Lemma 10.12 it suffices to bound, for each $k < q$, the randomness whose first bad step is k. Fix $\Delta \neq 0$ and inject it into the simulated randomness by Lemma 10.13. The images with $y = X + \Delta$ occur, for each simulated randomness, for at most one Δ, since the simulated game does not use Δ. The others lie, for every Δ, in the set of Lemma 10.15. Summing over Δ, the count is at most $|\mathrm{Sim}| + (|V|-1)|\mathrm{Sim}|/|V|$; divide by $(|V|-1)|\mathrm{Sim}|$ and sum over k. □

Theorem 10.17 (Adaptive authenticity)**.** *In the adaptive authenticity game, an adversary taking q actions wins with probability at most*

$$\frac{q+1}{|V|-1} + \frac{q}{|V|}.$$

For $|V| \geq 2$ this is at most $(2q+1)/(|V|-1)$, an elementary inequality that is not part of the formal statement.

Proof. Theorem 10.7 and Theorem 10.16. □

Remark 10.18 (The Chou–Orlandi OT). Theorem 10.17 is used in Chapter 11, with Lemma 8.9 (both labels of an input require solving square Diffie–Hellman) and a simulation of the OT layer by label requests, to extend the soundness of Chapter 7 to the Chou–Orlandi OT. The OT derives its keys with a hash as well. Theorem 10.17 is for an oracle used only by the garbling, so the two must hash in separate domains (for instance with distinct prefixes), which makes them independent random oracles.

Chapter 11

Soundness with the Chou–Orlandi OT

Status. Everything in this chapter is formalized, and no originality is claimed. The bound is a reduction: the hardness of square Diffie–Hellman is not assumed in Lean, and the efficiency of the solvers is outside the formalization, as for the DDH distinguishers of Chapter 9. It assembles three formalized results: the binding bound of Theorem 7.11, used as in the end-to-end argument of Theorem 7.15, adaptive authenticity (Theorem 10.17) and the difference of the two key points (Lemma 8.9). The new ingredient is a simulation of the OT layer by label requests (Lemma 11.6), and the reduction of the event that the prover queries both key points of an input to square Diffie–Hellman (Lemma 11.10). Both are proved in full below.

11.1 The game

Definition 11.1 (Soundness game with the Chou–Orlandi OT)**.** The group $\mathbb{G}$ has prime order q and g is a generator. The verifier samples, for each input $j < k$, a secret $a_j \in \mathbb{Z}_q$ and sends $A_j = a_j g$; the seed (Δ, L, L_c), with Δ uniform in $V \setminus \{0\}$; and three independent random oracles: the hash H of the garbling, the key hash $H_K(j, \cdot) : \mathbb{G} \to V$ of the OT (separated from H by its domain), and the commitment oracle G. The cheating prover

1. sees A, makes n_0 queries to H_K, and sends B_j for every j;
2. sees the material and the masked label pairs $e_j = (L_j + H_K(j, a_j B_j),\ L_j + \Delta + H_K(j, a_j(B_j - A_j)))$, makes n_1 steps, each with one query to H, H_K and G, and commits c;
3. sees the seed and the secrets a_j, makes n_2 steps, and opens (z, r).

The verifier accepts if z is the 1-label of the output and $G(z, r) = c$.

Theorem 11.2 (Soundness with the Chou–Orlandi OT, expected form)**.** *For an unsatisfiable circuit, the verifier accepts with probability at most*

$$\frac{Q+1}{|V|-1} + \frac{Q}{|V|} + \frac{k}{q} + \sum_{j<k} \sum_{i \neq i' < n_K} \mathrm{Adv}_{\mathrm{sqDH}}(S_{j,i,i'}) + \frac{n_1^2 + n_2 + 1}{|D|},$$

where $Q = 2(k + n_1)$ is the number of actions of the adversary of Lemma 11.6 in the adaptive game, $n_K = n_0 + n_1$ is the number of key queries, and $S_{j,i,i'}$ are explicit square Diffie–Hellman

solvers built from the prover (Definition 11.8). A solver depends on the commitment oracle G and on Δ, and $\mathrm{Adv}_{\mathrm{sqDH}}(S_{j,i,i'})$ is the probability, over a uniform $\alpha \in \mathbb{Z}_q$ and the coins of the solver, that it outputs $\alpha^2 g$ on input αg, averaged over G and $\Delta \neq 0$. The advantages are added, not averaged, over the $k\, n_K(n_K - 1)$ solvers.

Proof. Binding. Fix (a, H_K, J_K, ρ). Then the prover is a prover for the commitment oracle G alone, as in Theorem 7.15: it answers its queries to H and H_K itself. Its acceptance costs at most the probability, over G, of the event Ext that the first query of step 2 to G answered with the commitment is at the target, plus $(n_1^2 + n_2 + 1)/|D|$ (Theorem 7.11).

The OT layer. Fix G. The event Ext is contained in Zero$\cup$(Both$\setminus$Zero)$\cup$(Ext$\setminus$(Zero$\cup$Both)). The first has probability at most k/q, since each a_j is uniform. The second is bounded by Corollary 11.11, and the third by Corollary 11.7.

Averaging the first bound over (a, H_K, J_K, ρ) and the second over G gives the claim. □

11.2 The OT layer

Throughout this section G is fixed, and so are the circuit, the prover and the numbers of queries; the circuit is unsatisfiable, so its output has value 0 on every input and the target label $Z_{\mathrm{out}} + \Delta$ is the inactive output label for every input. We use the adaptive game of Definition 10.4: its randomness is $\rho = (L, L_c, H, J)$, with $\Delta \neq 0$ a separate parameter, its adversary sees the real material and then takes Q actions, each a query to H or a label request, and only the first request of an input counts.

Definition 11.3 (Key points and early queries)**.** In a run of the prover, let B_j be its message for input j. The *key points* of j are $K_j^0 = a_j B_j$ and $K_j^1 = a_j(B_j - A_j)$; they differ by $a_j A_j = a_j^2 g$ (Lemma 8.9), so they are distinct when $a_j \neq 0$. A query to H_K is *at* K_j^c if it is the point (j, K_j^c); the key hash is separated by the index j, so a query (j', K_j^c) with $j' \neq j$ is not at a key point of j. The key point K_j^c is *early* (Pre_j^c) if some query of step 1, before B is sent, is at it. We write Zero for the event that some $a_j = 0$, and Both for the event that, for some j, the prover queries both K_j^0 and K_j^1 before it commits.

Definition 11.4 (The adversary $\mathcal{A}_\gamma$)**.** The coins are $\gamma = (a, H'_K, e')$ with $a \in \mathbb{Z}_q^k$, $H'_K : [k] \times \mathbb{G} \to V$ and $e' \in (V \times V)^k$. The adversary $\mathcal{A}_\gamma$ of the adaptive game runs the prover as follows.

1. It sends $A_j = a_j g$, answers the queries of step 1 with H'_K, and obtains B; it computes the key points and the early ones (it knows a).
2. For each j and $c \in \{0, 1\}$: if K_j^c is early, it requests the bit c of input j, receives X, and puts $e_{j,c} = X + H'_K(j, K_j^c)$; otherwise $e_{j,c} = e'_{j,c}$. It gives the prover the material of the game and e.
3. At each step of step 2 it forwards the query to H to the game, answers the query to G with G, and answers the query (j, P) to H_K as follows. If $P = K_j^c$, K_j^c is not early, and this is the first query at K_j^c, it requests the bit c of input j, receives X, and answers $e_{j,c} + X$; a later query at the same point gets the same answer. Every other query gets $H'_K(j, P)$.
4. Its guess is the first component of the first query of step 2 to G whose answer is the commitment, and 0 if there is none.

It makes n_1 queries to H and at most $2k$ requests. In the formalization each input gets two actions before the masked pairs and each step two actions after them (a request or a query to H

outside the circuit, which answers 0 and is never bad), so it takes exactly $Q = 2(k + n_1)$ actions; a request is made at every query at a key point that is not early, not only the first, and the game answers repeated requests consistently.

The answer $e_{j,c} + X$ is the value of $H_K(j, K_j^c)$ that unmasks $e_{j,c}$ to X, since $e_{j,c} = X + H_K(j, K_j^c)$ in characteristic 2.

Definition 11.5 (The map to the coins)**.** The real randomness is $\omega = (a, H_K, J_K, \rho)$, where $J_K \in (V \times V)^k$ is junk that the game does not use (it only makes the two sides of the same size). Let e be the real masked pairs of the run of ω. The map Φ sends ω to (γ, ρ) with $\gamma = (a, H'_K, e')$, where, for every j and c such that K_j^c is not early, $H'_K(j, K_j^c) = J_{K,j,c}$ and $e'_{j,c} = e_{j,c}$, and, for every early K_j^c, $e'_{j,c} = J_{K,j,c}$; at all other points $H'_K = H_K$. Without Zero the two key points of an input are distinct, so these replacements do not collide.

Lemma 11.6 (The OT as label requests)**.** *Let ω be outside* Zero *and* Both*, and $\Phi(\omega) = (\gamma, \rho)$.*

1. (Same run.) *The run of the prover in the real game with ω, up to its commitment, is its run inside $\mathcal{A}_\gamma$ playing the adaptive game with ρ.*
2. (Injectivity.) *Φ is injective on the complement of* Zero $\cup$ Both*.*
3. (Wins.) *If, in the real run, the first query of step 2 to G answered with the commitment is at the target label, then $\mathcal{A}_\gamma$ wins the adaptive game with ρ.*

Proof. Step 1 of the prover. We show by induction on the queries of step 1 that the answers coincide. A query (j, P) is answered with $H_K(j, P)$ in the real game and with $H'_K(j, P)$ by $\mathcal{A}_\gamma$. These differ only if $P = K_j^c$ with K_j^c not early; but a query of step 1 at K_j^c makes K_j^c early. So the answers agree, the two runs of step 1 coincide, and $\mathcal{A}_\gamma$ computes the real B, the real key points and the real early ones.

The masked pairs. Fix j. Without Both, at most one of K_j^0, K_j^1 is early. If K_j^c is early, the request of $\mathcal{A}_\gamma$ for the bit c of j is the first request of input j, so the game answers $X = L_j + c\Delta$, and $e_{j,c} = L_j + c\Delta + H'_K(j, K_j^c) = L_j + c\Delta + H_K(j, K_j^c)$, the real value, since H'_K keeps the early key points. If K_j^c is not early, $e_{j,c} = e'_{j,c}$ is the real value by definition. The material of the adaptive game with ρ is the real material. So the prover receives the same messages.

Step 2 of the prover. By induction on the steps: the queries to H and to G are answered by the same oracles. Let (j, P) be a query to H_K. If $P = K_j^c$ with K_j^c not early, and it is the first query at that point, then no request of input j was made before: a request at the masked pairs would mean K_j^{1-c} is early, an earlier request in step 2 would mean a query at K_j^{1-c}, and either one, with the present query, is Both. So the game answers $X = L_j + c\Delta$, and $\mathcal{A}_\gamma$ answers $e_{j,c} + X = H_K(j, K_j^c)$, the real answer; a repeated query gets the same answer in both. Any other query is answered with $H'_K(j, P)$, which is $H_K(j, P)$ because P is not a replaced point. This proves the first claim.

Injectivity. Let $\Phi(\omega) = (\gamma, \rho)$. Then a and ρ are read off directly. Running the prover's step 1 with H'_K gives, by the first part, the real B, hence the key points and the early ones. Off the replaced points, $H_K = H'_K$. At a key point K_j^c that is not early, $H_K(j, K_j^c) = e'_{j,c} + L_j + c\Delta$ and $J_{K,j,c} = H'_K(j, K_j^c)$; at an early one, $J_{K,j,c} = e'_{j,c}$. So ω is determined by $\Phi(\omega)$.

Wins. By the first claim the guess of $\mathcal{A}_\gamma$ is the first component of the first query of step 2 to G answered with the commitment, the same as in the real run, so it is the target label. The target is $Z_{\text{out}} + \Delta$, and the active output label is Z_{out} for every input, since the output value is 0 on every input. So the guess is the active output label plus Δ for the input that $\mathcal{A}_\gamma$ requested, which is a win. □

Corollary 11.7 (The OT layer, counting form)**.** *For fixed G, summed over $\Delta \neq 0$, the real randomness for which the first query of step 2 to G answered with the commitment is at the target, outside* Zero ∪ Both*, is at most the sum over γ of the randomness ρ for which $\mathcal{A}_\gamma$ wins. By Theorem 10.17, with Δ and ω uniform, its probability is at most $(Q+1)/(|V|-1)+Q/|V|$.*

Proof. For each Δ, Φ maps this set injectively (Lemma 11.6) into the pairs (γ, ρ) with the same Δ for which $\mathcal{A}_\gamma$ wins. The real randomness and the pairs have the same number of elements, since (H_K, J_K) and (H'_K, e') do. For each γ, Theorem 10.17 bounds the sum over Δ of the winning ρ. □

11.3 Both key points

We bound the event Both input by input. Fix j. The key queries of the prover are numbered $0, 1, \ldots, n_K - 1$ in the order they are made, through steps 1 and 2, up to the commitment ($n_K = n_0 + n_1$); a key query is any query to H_K, whatever its index. For the event Both_j, let i and i' be the numbers of the first key queries at (j, K_j^0) and at (j, K_j^1); we write $\text{Both}_{j,i,i'}$ for this event. Without Zero the two key points are distinct, so $i \neq i'$, and the sums below run over the pairs $i \neq i'$. In this section $\Delta \neq 0$ and the commitment oracle G are fixed; the solvers depend on them.

Definition 11.8 (The solver $S_{j,i,i'}$)**.** The coins are $\sigma = (a_{-j}, \rho, H_K^*, e^*, J^*)$: the secrets of the other inputs, the randomness ρ of the garbling, a key hash $H_K^* : [k] \times \mathbb{G} \to V$, a pair $e^* \in V \times V$, and junk $J^* \in (V \times V)^{k-1}$ that it does not use. In the formalization the coins have the type of the real randomness, with $a_j = 0$ and e^* stored as the junk of j. Given $A \in \mathbb{G}$, the solver $S_{j,i,i'}$ runs the verifier and the prover, with $A_j = A$ and $A_{j'} = a_{j'} g$ for $j' \neq j$, as follows.

1. It garbles with ρ. For $j' \neq j$ it computes the masked pair $e_{j'}$ as the verifier does, with H_K^*, since it knows $a_{j'}$. For j it sends $e_j = e^*$.
2. It answers the key queries in order. The query number i, at a point (j_i, P_i), gets $e_0^* + L_j$, and the query number i', at $(j_{i'}, P_{i'})$, gets $e_1^* + L_j + \Delta$. A later query at the same point as query i (resp. i') gets the same answer. Every other key query (j'', P) gets $H_K^*(j'', P)$. It answers H with the H of ρ and G with G.
3. It outputs $P_i - P_{i'}$; the prover makes exactly n_K key queries before it commits, so both points exist for $i, i' < n_K$.

The solver does not use a_j: it only needs A, through the prover's view.

In the language of Theorem 8.10, for fixed σ the points $A \mapsto P_t$ of the key queries of this simulated run form a receiver, and $S_{j,i,i'}$ is its solver for the pair (i, i'). The receiver depends on (i, i'), because the answers to the queries i and i' do.

Definition 11.9 (The map to the solver's coins)**.** Let $\omega = (a, H_K, J_K, \rho)$ be outside Zero, with real masked pairs e and key points K_j^0, K_j^1. The map Ψ_j sends ω to (a_j, σ) with $\sigma = (a_{-j}, \rho, H_K^*, e^*, J^*)$, where $e^* = e_j$, $J^* = (J_{K,j'})_{j' \neq j}$, and H_K^* is H_K with its values at (j, K_j^0) and (j, K_j^1) replaced by $J_{K,j,0}$ and $J_{K,j,1}$.

Lemma 11.10 (Both key points)**.** *Let ω be outside* Zero*, in* $\text{Both}_{j,i,i'}$*, and $\Psi_j(\omega) = (a_j, \sigma)$.*

1. (Same run.) *The run of the prover in the real game with ω, up to its commitment, is its run inside $S_{j,i,i'}$ with coins σ and input $A = a_j g$.*

2. (Injectivity.) Ψ_j *is injective on* $\mathrm{Both}_{j,i,i'} \setminus \mathrm{Zero}$.

3. (Wins.) $S_{j,i,i'}$ *with coins* σ *wins on* a_j*: it outputs* $a_j^2 g$.

Proof. Same run. The material and the masked pairs agree: the solver garbles with the real ρ; for $j' \neq j$ it computes $e_{j'}$ with H_K^*, which agrees with H_K at index j'; and $e^* = e_j$. The answers to H and G agree. We show by induction on the key queries that their answers agree. Let the query number t be at (j'', P).

- If $t = i$: the point is (j, K_j^0), and the real answer is $H_K(j, K_j^0) = e_{j,0} + L_j$, which is the solver's answer $e_0^* + L_j$. Likewise for $t = i'$, with $H_K(j, K_j^1) = e_{j,1} + L_j + \Delta$.
- If t is later than i and the point is (j, K_j^0), both answers are those of the query i; likewise for i'.
- Otherwise the point is not a key point of j: an earlier query there would contradict the choice of i or i' as the first ones. So $H_K^*(j'', P) = H_K(j'', P)$, the real answer.

Since the answers agree, so do the queries, including B, which the prover computes from them.

Wins. By the first part, $P_i = K_j^0$ and $P_{i'} = K_j^1$, so the output is $K_j^0 - K_j^1 = a_j B_j - a_j(B_j - A_j) = a_j A_j = a_j^2 g$ (Lemma 8.9).

Injectivity. Let $\omega \in \mathrm{Both}_{j,i,i'} \setminus \mathrm{Zero}$. From (a_j, σ) we read a, ρ, $e_j = e^*$ and $J_{K,j'}$ for $j' \neq j$. Running $S_{j,i,i'}$ with coins σ on $A = a_j g$ gives, by the first part, the real run, hence B, K_j^0 and K_j^1, which are distinct since $a_j \neq 0$. Then $H_K = H_K^*$ off these two points, $H_K(j, K_j^0) = e_0^* + L_j$, $H_K(j, K_j^1) = e_1^* + L_j + \Delta$, and $J_{K,j,c} = H_K^*(j, K_j^c)$. So ω is determined. □

The first part uses $\mathrm{Both}_{j,i,i'}$ only through the choice of i and i' as the first queries at the key points. The injectivity uses the first part, so it holds on $\mathrm{Both}_{j,i,i'} \setminus \mathrm{Zero}$ for fixed i and i', and not on the whole complement of Zero.

Corollary 11.11 (Both key points, counting form)**.** *With* ω *uniform,*

$$\Pr[\mathrm{Both} \wedge \neg\mathrm{Zero}] \leq \sum_{j<k} \sum_{i \neq i' < n_K} \Pr_{a_j, \sigma}[S_{j,i,i'} \text{ wins on } a_j],$$

where a_j *is uniform in* $\mathbb{Z}_q$ *and* σ *uniform, independent: the right side is the sum of the square Diffie–Hellman advantages of the solvers.*

Proof. Both is the union of the $\mathrm{Both}_{j,i,i'}$, and each of them, outside Zero, injects by Ψ_j into the pairs (a_j, σ) on which $S_{j,i,i'}$ wins (Lemma 11.10). The real randomness and the pairs have the same number of elements: (H_K, J_K) corresponds to (H_K^*, e^*, J^*). □

Remark 11.12 (The two points that need care). The case $a_j = 0$ is a genuine leak of the protocol as modelled: with $A_j = 0$ the two masks coincide. It costs k/q, which is negligible, but it must appear in the bound. The second point is that the square Diffie–Hellman reduction cannot recognize the key queries, which is why the solvers guess their positions, at a cost of $n_K(n_K - 1)$ solvers per input.

Chapter 12

Ingredients for adaptive corruptions

Status. Everything in this chapter is formalized, and no originality is claimed. Section 2.5 describes what a simulator must explain when the prover is corrupted during or after the protocol of [JKO13]. This chapter formalizes some protocol-specific facts behind that description. The equivocation condition of Definition 12.1 is the condition on the receiver's message and randomness in [GKPS18, Sect. 3, Definition 3.1], in a perfect version; the rest of that definition (a two-round OT that securely realizes the OT functionality in the common reference string model, and a simulator that uses only the trapdoor of the setup) is not formalized, and neither is DDH. The equivocation of the PVW receiver in decryption mode is the equivocation step of the proof of [GKPS18, Theorem 3.2], and the explanation of the prover follows [GKPS18, Sect. 3.2]. The other statements we have not found published as such: that the explained PVW secret decrypts, and that messy mode does not satisfy the condition, follow from the form of the keys; the Chou–Orlandi equivocation given the sender's secret follows from the form of the messages; the equivocation of the hash commitment applies the programming of the random oracle recalled in [GKPS18, Sect. 1.2] to the games of Chapter 7; and soundness in decryption mode combines Theorem 9.9 with the indistinguishability of the modes [PVW08, Theorem 5.2]. Adaptive security itself, which needs a model of execution with corruptions such as the one of universal composability [Can20, Sect. 7.1], is not defined here: the theorems below are some ingredients of a simulation, not the simulation. Among the missing ingredients are the consistency of the Chou–Orlandi outputs with an explained message, the composition of the explanations of the OT and of the commitment, zero knowledge with the PVW OT in decryption mode, and extraction against a corrupted prover.

12.1 The equivocation condition for the receiver

Definition 12.1 (Equivocation condition)**.** Let the receiver of an OT send $msg(\sigma, r)$ for the choice bit σ and the randomness r. It satisfies the *equivocation condition*, with simulated message sim and explanations e_0, e_1, if each e_σ is a bijection of the randomness and

$$msg(\sigma, e_\sigma(r)) = sim(r) \quad \text{for all } \sigma, r.$$

A simulator sends $sim(r)$ for a uniform r, without the choice bit; told σ later, it explains its message by the randomness $e_\sigma(r)$, which is uniform because e_σ is a bijection. This is the

condition on messages and randomness of [GKPS18, Sect. 3, Definition 3.1], where the explained randomness is only required to be statistically close to the honest one; here it has exactly the same distribution. That definition also asks for a two-round OT that securely realizes the OT functionality in the common reference string model, with explanations computed by a simulator from the trapdoor of the setup; these requirements are not part of Definition 12.1, and who can compute the explanations is said for each instance below.

Theorem 12.2 (The PVW receiver in decryption mode)**.** *On a decryption-mode common string, with $g_1 = y \cdot g_0$, $y \neq 0$ and $h_b = x \cdot g_b$, the PVW receiver satisfies the equivocation condition with simulated message $keyGen(0, r)$, $e_0(r) = r$ and $e_1(r) = y^{-1}r$; the explanation uses the trapdoor y of the decryption setup. Moreover, for any explanations satisfying the condition, the explained secret $e_\sigma(r)$ decrypts what the sender encrypts on branch σ under the simulated key.*

Proof. By Definition 9.6, $keyGen(1, r) = keyGen(0, yr)$, so $keyGen(1, y^{-1}r) = keyGen(0, r)$; multiplication by y^{-1} is a bijection of $\mathbb{Z}_q$. This is the equivocation step of the proof of [GKPS18, Theorem 3.2] and the algorithm TrapKeyGen of [PVW08, Sect. 5.3]. The simulated key is the key of the choice σ with secret $e_\sigma(r)$, so the second claim is the correctness of decryption (Lemma 9.2). □

Proposition 12.3 (Not in messy mode)**.** *On a messy common string the PVW receiver does not satisfy the equivocation condition, for any simulated message and explanations.*

Proof. If $keyGen(0, r_0) = keyGen(1, r_1)$, then $r_0 g_0 = r_1 g_1$ and $r_0 x_0 g_0 = r_1 x_1 g_1$, so $(x_0 - x_1) r_0 g_0 = 0$ and $r_0 = 0$, since $x_0 \neq x_1$ and $g_0 \neq 0$. Hence e_0 is identically 0, and it is not a bijection. □

In our reading, this is why the instantiation of [GKPS18, Sect. 3.3] uses decryption mode: in messy mode the key determines the branch, which is what makes extraction possible (Lemma 9.4).

Proposition 12.4 (The Chou–Orlandi receiver, given the sender's secret)**.** *Let $A = a \cdot g$. The receiver message $B = b \cdot g + wA$ satisfies the equivocation condition with simulated message $b \cdot g$ and explanations $e_0(b) = b$, $e_1(b) = b - a$.*

Proof. $(b - a) \cdot g + a \cdot g = b \cdot g$, and $b \mapsto b - a$ is a bijection. □

The explanation needs the sender's secret a, which is not a trapdoor of a common string, so this does not make the Chou–Orlandi OT receiver equivocal in the sense of [GKPS18, Definition 3.1]. The simulator knows a when it plays the honest verifier, and otherwise learns it only at the open-all step (Section 2.5).

12.2 Explaining the prover

Theorem 12.5 (Explaining the prover)**.** *Let the OT receiver satisfy the equivocation condition, and let the verifier choose its message (the garbled circuit, the OT inputs and the seed it opens) as a function of the prover's OT messages. Let $C(w) = 1$, and let r_j be the simulator's randomness for the j-th OT. With the explained randomness $e_{w_j}(r_j)$ the prover's OT messages are the simulated ones $sim(r_j)$, and the verifier's view of the prover is the output of the simulator of Theorem 4.5. The explained randomness is uniform when r is. Here only the receiver's first message is a real OT message: the delivery of the labels is the ideal one of Chapter 4.*

Proof. The first claim is the definition, coordinate by coordinate. Then the verifier's messages are the same in both cases, and Theorem 4.5 gives the second claim. The explanation is a bijection in each coordinate. □

According to [GKPS18, Sect. 3.2], with an ideal commitment and an OT that is receiver equivocal in their sense, the prover's view of the OT is all a simulator must explain when the prover is corrupted after the execution, since every step of the prover after the oblivious transfers is independent of the witness. Theorem 12.5 proves the part of this that concerns the receiver's messages and the verifier's view, with ideal delivery of the labels; for the PVW OT, Theorem 12.2 adds that the explained secret decrypts the label of the actual choice.

12.3 Equivocating the hash commitment

With the hash commitment of Chapter 7, a prover corrupted after committing and before opening hands over its output label Z_w and its commitment randomness r. In the real execution its commitment is $G(Z_w, r)$. The simulator commits instead to a uniform c, without the witness, and at the corruption reveals (Z_w, r) and sets $G(Z_w, r) := c$; this is possible in the *programmable* random oracle model [GKPS18, Sect. 1.2]. We reuse the games of Definition 7.3, with the prover's check of the opened seed removed (it always passes), so that the last message of the real game is always (Z_w, r). The corruption thus takes place after the verifier has sent its seed, and the state revealed is only (Z_w, r); the prover's OT messages are handled by Theorem 12.5, and the two explanations are not composed here.

Theorem 12.6 (Equivocation by programming the oracle)**.** *Let a malicious verifier make q_1, q_2 and q_3 oracle queries before the commitment, between the commitment and the corruption, and after the corruption. Over G, c and r uniform, the probabilities that it outputs* 1 *in the real execution and against the simulator differ by at most $(q_1 + q_2)/|R|$.*

Proof. Off the bad event Bad_{23} of Lemma 7.6, the hybrid of Lemma 7.5, which programs the oracle already before the second phase, and the simulator differ only in the answers of the second phase, and these do not hit the programmed point (Lemma 7.2). Since the check always passes, the last message is never an abort, and the bad event reduces to a query of the second phase at the commitment point; the queries of the third phase are answered by the programmed oracle in both games and cost nothing. The rest is the counting of Theorem 7.7, applied to the verifier without its third phase. □

12.4 Soundness in decryption mode

The PVW receiver satisfies the equivocation condition in decryption mode, while the soundness of Theorem 9.9 uses messy mode, where the sender is statistically protected. Switching the modes costs the DDH advantage of the acceptance probability.

Theorem 12.7 (Soundness with the PVW OT in decryption mode)**.** *Let C be unsatisfiable, $q > 2$, and let the cheating prover depend on the common string and make at most N_1 and N_2 queries in its two phases. Averaged over the decryption-mode setup, the verifier accepts with probability at most*

$$\frac{N_1 + 1}{|V| - 1} + \frac{N_1^2 + N_2 + 1}{|D|} + \mathrm{Adv}_{\mathrm{DDH}}(p_{\mathrm{acc}}) + \frac{3}{q},$$

where p_{acc} is the acceptance probability as a function of the common string. The bound holds on average over the setup, not for each string; as in Theorem 9.20, the efficiency of the distinguisher p_{acc} is outside the formalization.

Proof. The strings of the messy setup are messy, so by Theorem 9.9 the acceptance probability on each of them is at most the first two terms, and so is its average. The acceptance probability lies in $[0, 1]$, and Theorem 9.19 bounds the difference of the two averages. □

Chapter 13

Cut-and-choose against a malicious garbler

When the garbling cannot be opened, for instance because it contains a secret of the garbler, a malicious garbler is handled by cut-and-choose: the garbler sends N circuits, the evaluator opens some of them and evaluates the others. Classical analyses take a majority vote among the evaluated circuits. Lindell [Lin13] observed that if *one good evaluated circuit suffices*, choosing each circuit independently with a fair coin gives error 2^{-N}. The first section contains the combinatorial core; the second connects it with the garbling of Chapter 3.

Definition 13.1 (Cheating)**.** Let $B \subseteq \{0, \dots, N-1\}$ be the set of bad circuits and E the set of evaluated circuits. The garbler cheats if no opened circuit is bad (it is not caught) and no evaluated circuit is good (there is no correct output).

Lemma 13.2. *The garbler cheats if and only if $E = B$.*

Proof. The first condition says $B \subseteq E$, the second $E \subseteq B$. □

Theorem 13.3 (A fixed number of evaluated circuits)**.** *If E is uniform among the subsets of size $t \leq N$, the garbler cheats with probability at most $1/\binom{N}{t}$, with equality when $|B| = t$.*

Proof. By Lemma 13.2 at most one t-subset lets the garbler cheat, and exactly one if $|B| = t$. □

Theorem 13.4 (Independent fair coins)**.** *If each circuit is evaluated or opened with an independent fair coin, every garbler cheats with probability exactly 2^{-N}.*

Proof. E is uniform among the 2^N subsets, and exactly one of them is B. □

Proposition 13.5. $\binom{N}{t} \leq 2^N$. *Independent coins are never worse than a fixed number of evaluated circuits.*

Remark 13.6. The gap is small. For $N = 40$ the best fixed $t = 20$ gives $2^{-37.0}$ against 2^{-40}, and for $N = 96$ it gives $2^{-92.4}$ against 2^{-96}. With independent coins the evaluation set may be empty. Here that event is counted inside the bound 2^{-N}; Lindell excludes it (the set of check circuits must not be all of them), so his error is $1/(2^s - 1)$, which he rounds to 2^{-s} [Lin13, Theorem 4.1 and its footnote 3]. The argument that cheating requires every bad circuit to be evaluated and every good one to be checked is in [Lin13, Sect. 2].

13.1 Copies of a garbling

The two hypotheses behind Definition 13.1, that an opened bad circuit is caught and that a good evaluated circuit produces the correct output, follow from the garbling of Chapter 3. How the evaluator chooses among the outputs of several evaluated circuits, some of them bad, is not modeled.

Definition 13.7 (Copies)**.** A copy consists of a claimed seed, the material and the pair of labels offered for each input. It is good when it passes the check of Definition 4.2: its material is the garbling of C with its seed and its pairs are $(L_j, L_j + \Delta)$. Opening a copy reveals the seed, so goodness is checked by garbling again.

Lemma 13.8 (One good evaluated copy suffices)**.** *A good copy evaluated on x outputs the plaintext value $C(x)$ and the label of that value.*

Proof. Theorem 3.8. □

Theorem 13.9 (Caught or correct)**.** *For every evaluation set E, an opened copy is bad, or an evaluated copy outputs $C(x)$ with its label, or E is exactly the set of bad copies. The last event has probability at most $1/\binom{N}{t}$ for a uniform t-subset, and exactly 2^{-N} with independent fair coins.*

Proof. Lemma 13.2 with B the set of bad copies, Lemma 13.8, and Theorems 13.3 and 13.4. □

Remark 13.10. The evaluator receives its input labels through the offered pairs, as in an oblivious transfer. The consistency of the garbler's own inputs across the evaluated copies is not modeled.

Chapter 14

What is not formalized

We list the gaps between the statements above and a complete security proof.

- **The real OT.** Soundness and zero knowledge are proved with the PVW OT (Chapter 9); zero knowledge is reduced to DDH by explicit distinguishers. Lean has no notion of efficient computation, so the efficiency of these distinguishers, like the hardness of DDH itself, is outside the formalization. Soundness with the Chou–Orlandi OT is formalized as a reduction (Theorem 11.2): its bound contains the advantages of explicit square Diffie–Hellman solvers, whose efficiency, like the hardness of the problem, is outside Lean.
- **The hash.** Authenticity is proved with the hash as a random oracle. Outside that model, correlation robustness alone is not enough to prove Free XOR secure, in the sense that a fully black-box reduction is ruled out [CKKZ12, Sect. 3]; Choi, Katz, Kumaresan and Zhou show that a circular form of it suffices [CKKZ12, Sect. 4, Definition 4 and Theorem 1].
- **Implementations.** The formalization is about the abstract garbling scheme of Chapter 3; no implementation of it is verified.
- **Number of queries.** The adversaries make a fixed number of queries, in a fixed pattern. Reducing an adversary that makes fewer queries, or interleaves them differently, to this form by padding with dummy queries is stated in the introduction but not formalized.
- **Adaptive corruptions and composition.** All security statements concern a single execution. Security against adaptive corruptions and universal composition are discussed in Section 2.5 but not formalized; Chapter 12 proves only some protocol-specific ingredients (the equivocation condition for the receiver, the explanation of the prover, the equivocation of the commitment, soundness in decryption mode), not a simulation in a model of execution with corruptions.
- **Cut-and-choose.** The consistency of the garbler's own inputs across the evaluated copies is not modeled.

Bibliography

[ABB+17] J. B. Almeida, M. Barbosa, G. Barthe, F. Dupressoir, B. Grégoire, V. Laporte and V. Pereira, *A fast and verified software stack for secure function evaluation*, ACM CCS 2017, 1989–2006. IACR ePrint 2017/821.

[BAG17] D. Butler, D. Aspinall and A. Gascón, *How to simulate it in Isabelle: towards formal proof for secure multi-party computation*, ITP 2017, LNCS, Springer, 114–130.

[BDZ03] F. Bao, R. H. Deng and H. Zhu, *Variations of Diffie–Hellman problem*, ICICS 2003, LNCS **2836**, Springer, 2003, 301–312.

[BHKO26] C. Barnum, D. Heath, V. Kolesnikov and R. Ostrovsky, *Adaptive garbled circuits and garbled RAM from non-programmable random oracles*, ITC 2026, LIPIcs **385**, 11:1–11:21. Full version: IACR ePrint 2023/1527.

[BHR12] M. Bellare, V. T. Hoang and P. Rogaway, *Adaptively secure garbling with applications to one-time programs and secure outsourcing*, ASIACRYPT 2012, LNCS **7658**, Springer, 2012, 134–153.

[BHR12b] M. Bellare, V. T. Hoang and P. Rogaway, *Foundations of garbled circuits*, ACM CCS 2012. Cited from the full version, IACR ePrint 2012/265.

[BLS20] D. A. Basin, A. Lochbihler and S. R. Sefidgar, *CryptHOL: game-based proofs in higher-order logic*, J. Cryptology **33** (2020), 494–566. Cited from IACR ePrint 2017/753.

[BPRS17] M. Byali, A. Patra, D. Ravi and P. Sarkar, *Fast and universally-composable oblivious transfer and commitment scheme with adaptive security*, IACR ePrint 2017/1165.

[BR06] M. Bellare and P. Rogaway, *The security of triple encryption and a framework for code-based game-playing proofs*, EUROCRYPT 2006, LNCS **4004**, Springer, 2006, 409–426. Cited from the full version, IACR ePrint 2004/331.

[BR93] M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, ACM CCS 1993, 62–73.

[Can20] R. Canetti, *Universally composable security*, J. ACM **67** (2020), no. 5, Article 28, 1–94.

[CCcore] CatCrypt Core, README of the public repository, https://github.com/spitters/CatCrypt-core, as of November 9, 2026.

[CKKZ12] S. G. Choi, J. Katz, R. Kumaresan and H.-S. Zhou, *On the security of the "free-XOR" technique*, TCC 2012, LNCS **7194**, Springer, 2012, 39–53.

[CO15] T. Chou and C. Orlandi, *The simplest protocol for oblivious transfer*, LATINCRYPT 2015. Cited from the revised version, IACR ePrint 2015/267 (2018).

[CSV19] R. Canetti, A. Stoughton and M. Varia, *EasyUC: using EasyCrypt to mechanize proofs of universally composable security*, IEEE CSF 2019. Cited from the extended version, IACR ePrint 2019/582.

[DFMS26] S. Dziembowski, G. Fabiański, D. Micciancio and R. Stefański, *Computationally sound symbolic cryptography in Lean*, IEEE CSF 2026, 620–635. Cited from IACR ePrint 2025/1700.

[DKLs18] J. Doerner, Y. Kondi, E. Lee and a. shelat, *Secure two-party threshold ECDSA from ECDSA assumptions*, IEEE S&P 2018, 980–997. Cited from the full version, IACR ePrint 2018/499 (2023).

[FNO15] T. K. Frederiksen, J. B. Nielsen and C. Orlandi, *Privacy-free garbled circuits with applications to efficient zero-knowledge*, EUROCRYPT 2015, Part II, LNCS **9057**, Springer, 2015, 191–219. Cited from the full version, IACR ePrint 2014/598.

[GIR] Z. A. Genç, V. Iovino and A. Rial, *"The simplest protocol for oblivious transfer" revisited*, IACR ePrint 2017/370; Information Processing Letters (2020).

[GKPS18] C. Ganesh, Y. Kondi, A. Patra and P. Sarkar, *Efficient adaptively secure zero-knowledge from garbled circuits*, PKC 2018, Part II, LNCS **10770**, Springer, 2018, 499–529.

[GYWYL] X. Guo, K. Yang, X. Wang, Y. Yu and Z. Liu, *Unmodified half-gates is adaptively secure — so is unmodified three-halves*, IACR ePrint 2023/1528, version of July 29, 2025.

[HJO+16] B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro and D. Wichs, *Adaptively secure garbled circuits from one-way functions*, CRYPTO 2016, Part III, LNCS **9816**, Springer, 2016, 149–178.

[HRV+23] P. G. Haselwarter, E. Rivas, A. Van Muylder, T. Winterhalter, C. Abate, N. Sidorenco, C. Hriţcu, K. Maillard and B. Spitters, *SSProve: a foundational framework for modular cryptographic proofs in Coq*, ACM Trans. Program. Lang. Syst. **45** (2023), no. 3, 1–61. Cited from IACR ePrint 2021/397.

[JKO13] M. Jawurek, F. Kerschbaum and C. Orlandi, *Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently*, ACM CCS 2013. Cited from the full version, IACR ePrint 2013/073.

[JW16] Z. Jafargholi and D. Wichs, *Adaptive security of Yao's garbled circuits*, TCC 2016-B, Part I, LNCS **9985**, Springer, 2016, 433–458.

[KS08] V. Kolesnikov and T. Schneider, *Improved garbled circuit: free XOR gates and applications*, ICALP 2008, Part II, LNCS **5126**, Springer, 2008, 486–498.

[Lin13] Y. Lindell, *Fast cut-and-choose based protocols for malicious and covert adversaries*, CRYPTO 2013; full version in J. Cryptology **29** (2016), 456–490. Locations are cited from the J. Cryptology version.

[Mas] P. Massot, *leanblueprint*, a plasTeX plugin to build formalization blueprints, https://github.com/PatrickMassot/leanblueprint.

[Mathlib20] The mathlib Community, *The Lean mathematical library*, CPP 2020, 367–381.

[MU21] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, CADE-28, LNAI **12699**, Springer, 2021, 625–635.

[PVW08] C. Peikert, V. Vaikuntanathan and B. Waters, *A framework for efficient and composable oblivious transfer*, CRYPTO 2008, LNCS **5157**, Springer, 2008, 554–571. Cited from the full version, IACR ePrint 2007/348, revised January 23, 2019; we also compared the version of August 25, 2008.

[Sho04] V. Shoup, *Sequences of games: a tool for taming complexity in security proofs*, IACR ePrint 2004/332.

[Spi26] B. Spitters, *CatCrypt: from Rust to cryptographic security in Lean*, IACR ePrint 2026/604.

[TDW+26] D. Tuma, Q. Dao, J. Waters, A. Hicks and N. Hopper, *VCVio: verified cryptography in Lean via oracle effects and handlers*, IACR ePrint 2026/899.

[Yao86] A. C. Yao, *How to generate and exchange secrets*, FOCS 1986, 162–167.

[ZRE15] S. Zahur, M. Rosulek and D. Evans, *Two halves make a whole: reducing data transfer in garbled circuits using half gates*, EUROCRYPT 2015, Part II, LNCS **9057**, Springer, 2015, 220–252.