Data Protection

Jorge L. Villar

MCYBERS, UPC, Fall 2025

Outline

Zero-Knowledge Proofs

A protocol between a prover P and a verifier V, such that:

- P and V have some common data (e.g., the statement to be proved)
- P has some private information (e.g., a witness that shows that the statement is true)
- P and V interact exchanging some messages
- P convinces V that the statement is true

Properties of a ZK Proof

- **Correctness:** An honest P with a valid witness for a statement is always accepted by an honest V.
- **Soundness:** A dishonest P cannot convince an honest V about a false statement.
- **Zero-Knowledge:** After a protocol execution with an honest P, V learns nothing but the fact that the statement is true.

  After the protocol, V cannot convince anybody else about the truthness of the statement.

Relaxations of Soundness

Statistical Soundness: The probability that an unbounded dishonest *P* convinces an honest *V* about a false statement is a neglibible function of some size parameter.

Bounded Soundness: The probability of cheating of an unbounded *P* is upper-bounded by some constant (e.g., 1/2).

The upper-bound can be arbitrarily reduced by independent repetition of the protocol.

Computational Soundness: Any efficient (i.e., a PPTM) dishonest *P* that can cheat an honest *V* with a negligible probability.

Zero-Knowledge as Simulatability

Zero-Knowledge is formalized as the existence of a simulator S that, from the interaction with V, it outputs a **transcript** similar to the corresponding one form a real execution of the protocol.

Transcript: Sequence of messages exchanged between the parties during a protocol execution.

Perfect Zero-Knowledge: The probability distributions of the real and simulated transcripts are identical.

Computational Zero-Knowledge: Any efficient algorithm can tell apart both distributions only with a negligible probability.

A related notion is **Witness Indistinguishability**: if there exist several witnesses for a given statement, V cannot learn which one was used by P.

Proofs of Knowledge

The statement of a Zero-Knowledge Proof typically has the form "I know a witness for this statement". To prove the soundness it suffices to show the existence of a **Knowledge Extractor**.

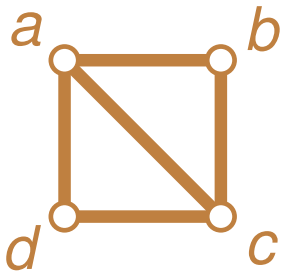
A Knowledge Extractor is an algorithm that extracts a valid witness form a (possibly dishonest) P that convinces an honest V with a probability above some threshold.

A typical Knowledge Extractor runs several instances of P until tow related transcripts are obtained, and from them the witness can be efficiently computed.

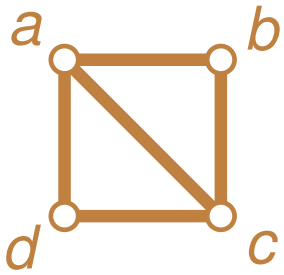
Example: Graph Isomorphism

A **graph** is a set of connected vertices. V is the set of vertices (points on a plane) and E is a set of "connections", i.e., pairs of vertices.

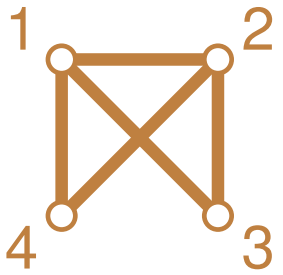
$V = \{a, b, c, d\}$, $E = \{ab, bc, cd, da, ac\}$



A **graph isomorphism** is a bijective map between graphs that preserves the connections.



$a \mapsto 1$
$b \mapsto 3$
$c \mapsto 2$
$d \mapsto 4$



Finding isomorphisms between large graphs is conjectured to be a hard problem.

Proof of Graph Isomorphism

P convinces V that the two graphs G_0 and G_1, with the same set of vertices, are isomorphic.

The witness of P is an isomorphism $\phi : G_0 \to G_1$.

- P chooses a random permutation of the vertices that defines a graph isomorphism $\psi_1 : G_1 \to G_2$, and also defines $\psi_0 = \psi_1 \circ \phi : G_0 \to G_2$.
- P sends G_2 to V.
- V selects a random bit $c \leftarrow \{0, 1\}$ and asks P for ψ_c.
- V accepts if ψ_c is bijective and $\psi_c(G_c) = G_2$.

1/2-Soundness

If P can convince V with probability $1/2 + \alpha$ for $\alpha > 0$ then G_0 and G_1 are isomorphic.

For some G_2, P is able to give a correct answer for both values of c.

If α is non-negligible, then a knowledge extractor can efficiently obtain an isomorphism $\phi : G_0 \to G_1$.

From two correct answers ψ_0, ψ_1 for the same G_2 the extractor can take $\phi = \psi_1^{-1} \circ \psi_0$.

Simulatability

A simulator can first choose $c' \leftarrow \{0, 1\}$, then take a random permutation η of the vertices and define $G_2 = \eta(G_{c'})$.

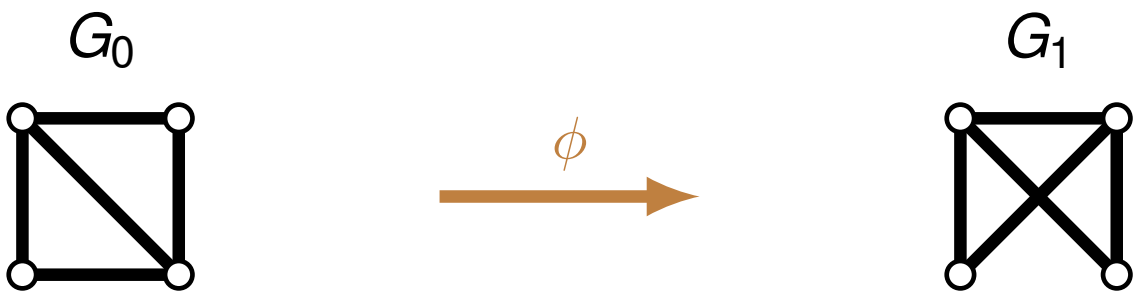
Then, the simulator sends G_2 to V. V chooses a random $c \leftarrow \{0, 1\}$.

If $c' \neq c$, then the simulator aborts the execution and restarts with a new random choice of c.

Otherwise, the simulator obtains the transcript (G_2, c, η), that is identically distributed as a real accepting transcript.

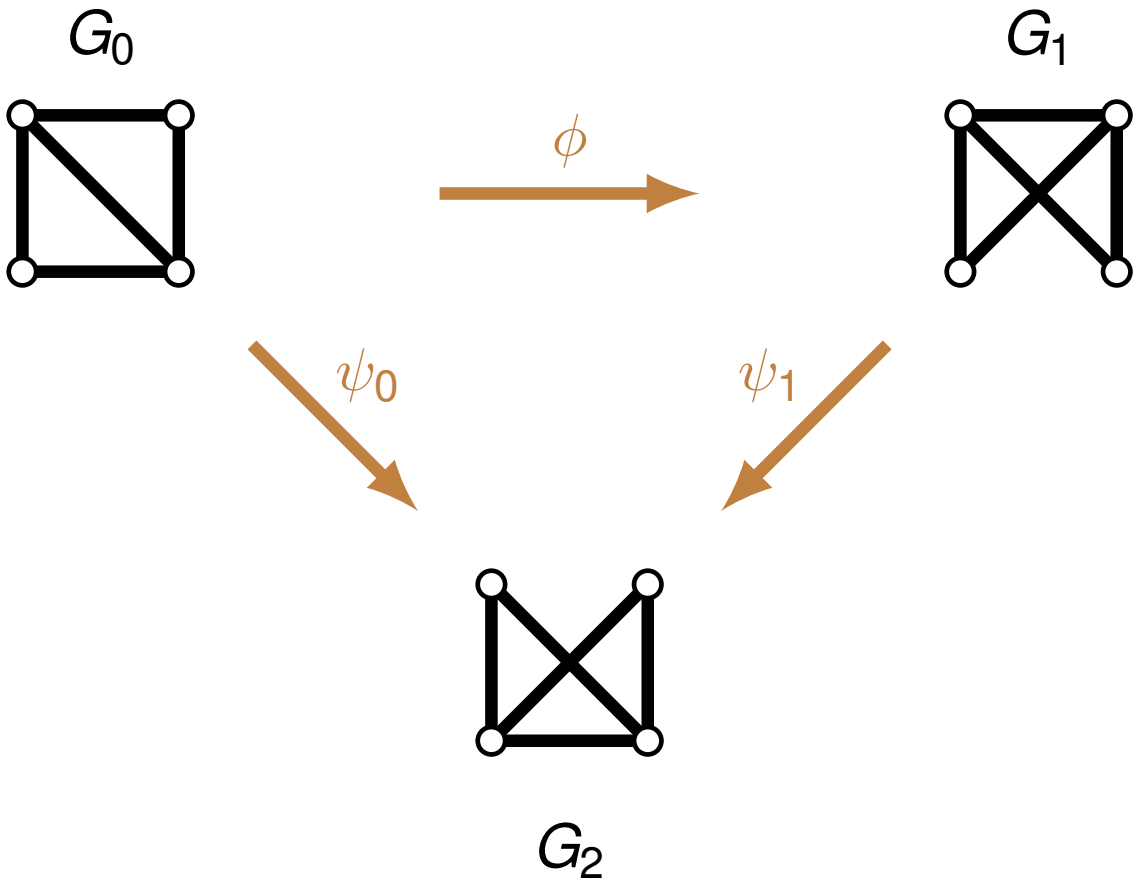
The simulator works even when V is dishonest and sends a biased choice of c.

Example



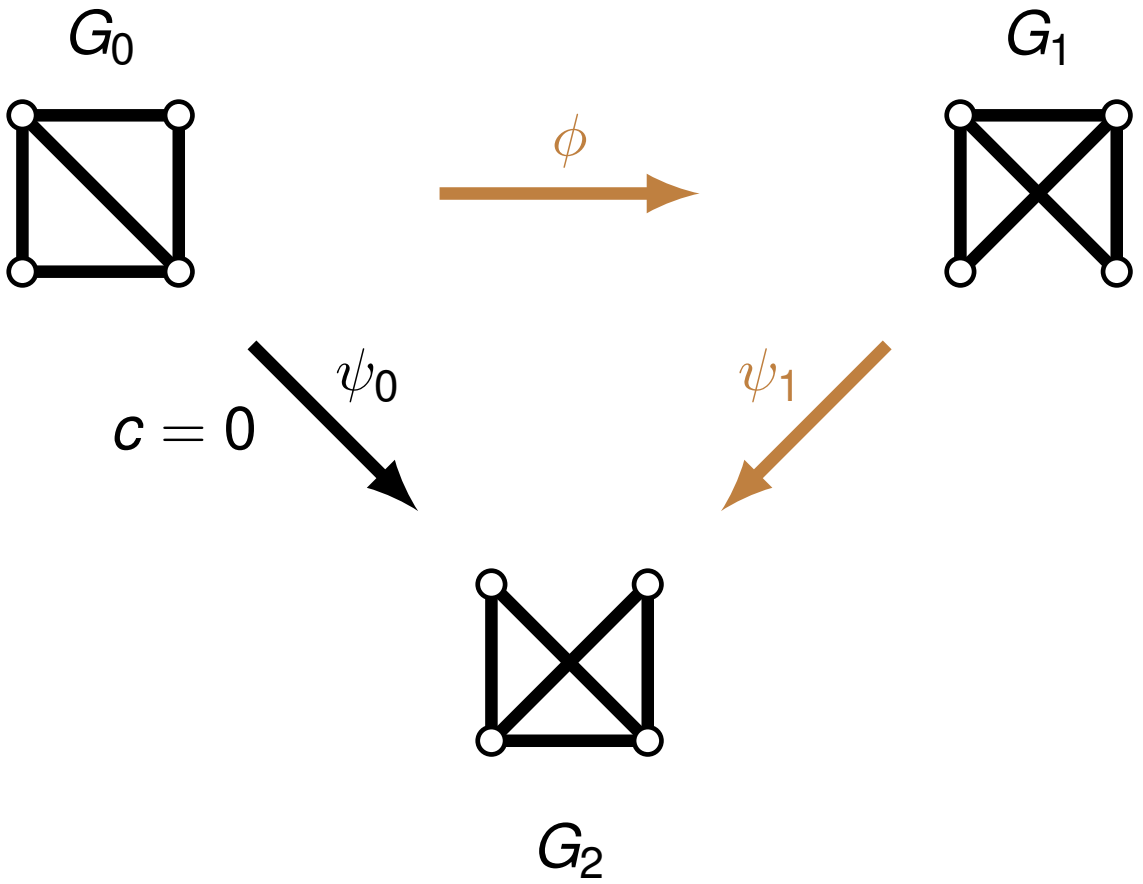
Initial state: P and V know G_0 and G_1, only P knows ϕ

Example



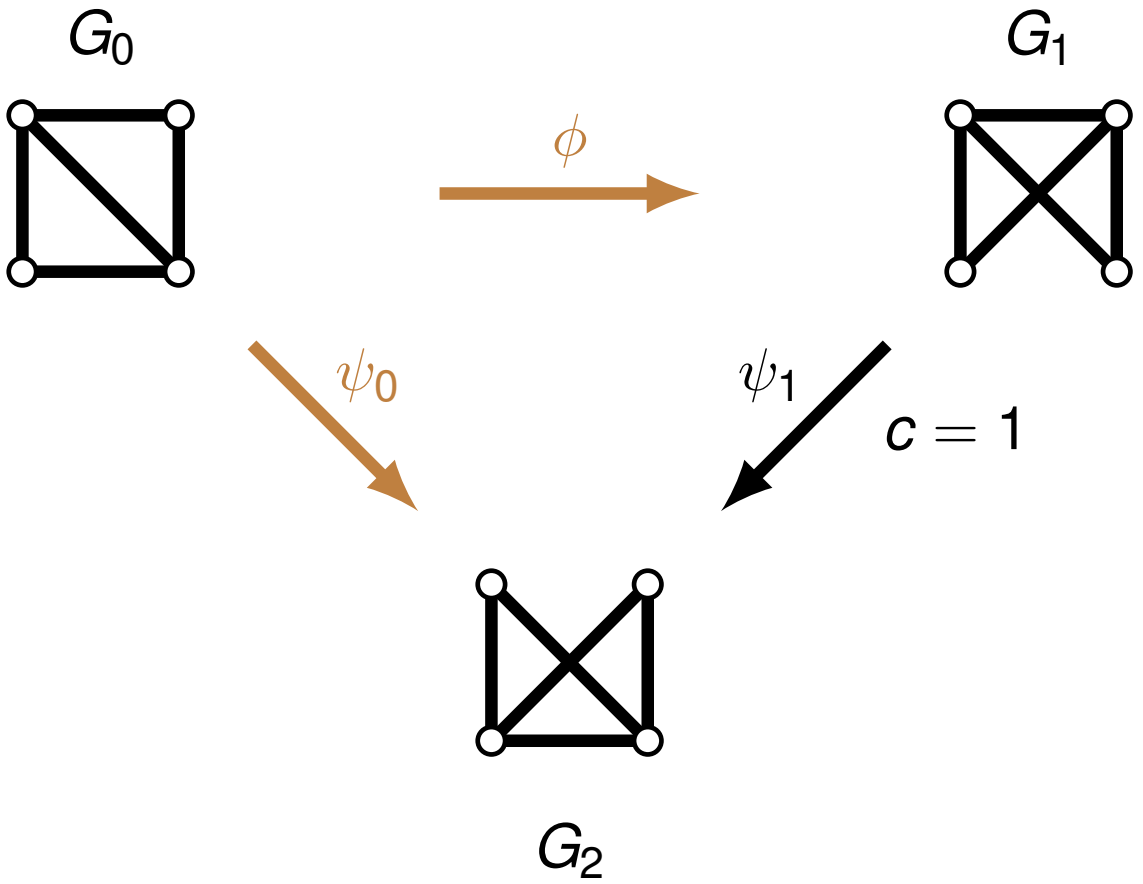
Commit: P reveals G_2 to V

Example



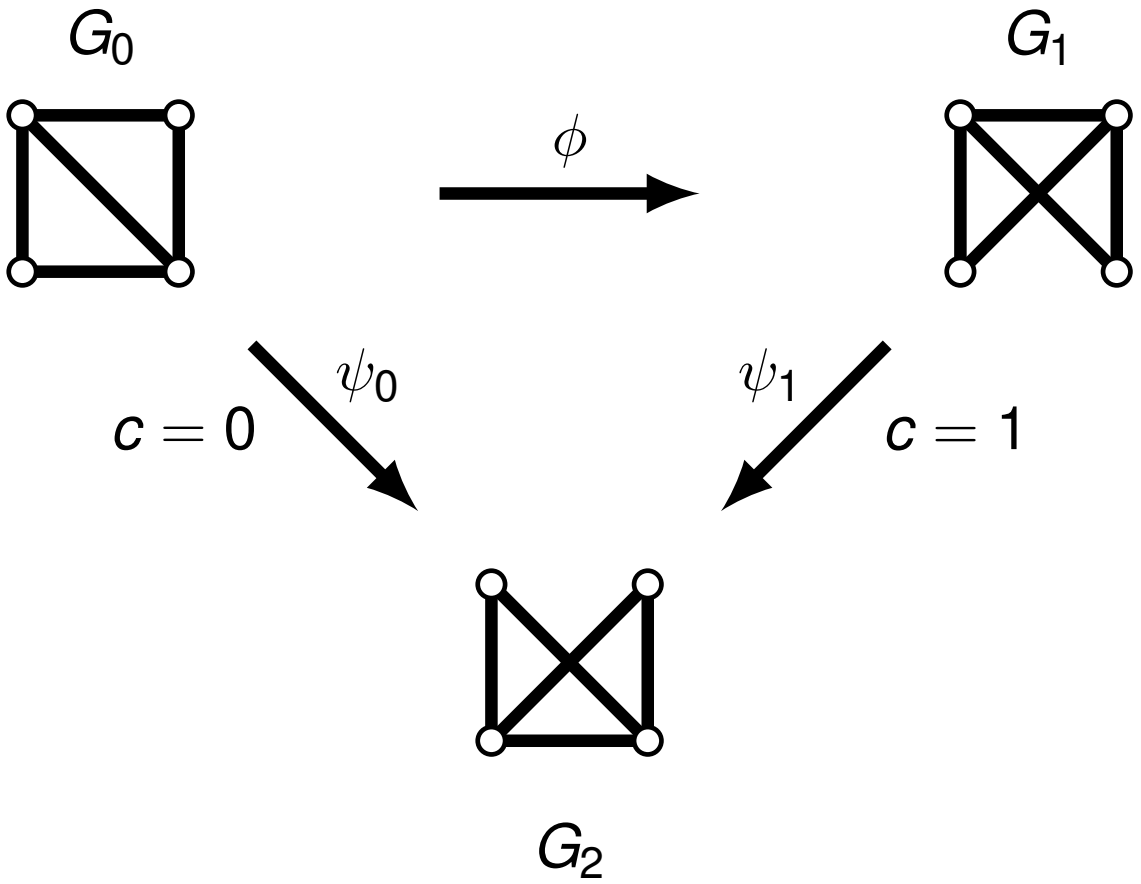
Respond: V asks for ψ_0, but ϕ remains hidden

Example



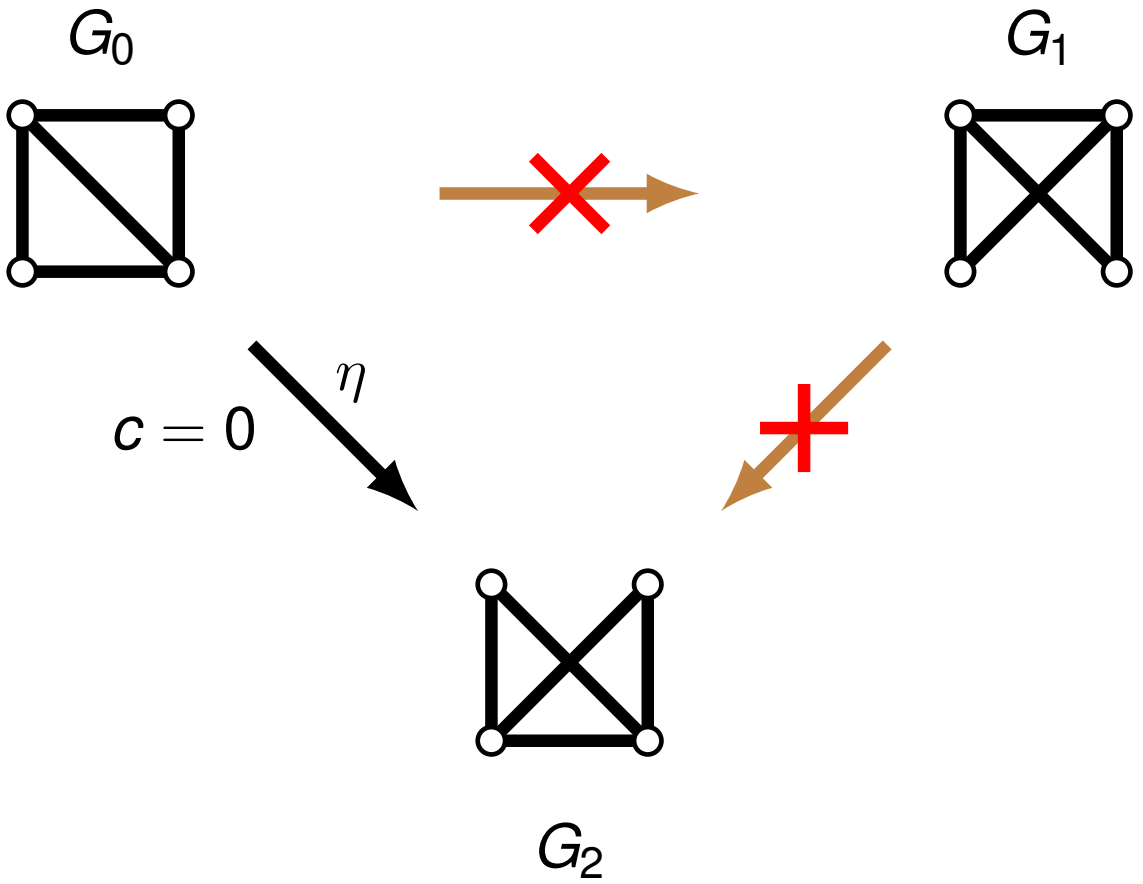
Respond: V asks for ψ_1, but ϕ remains hidden

Example



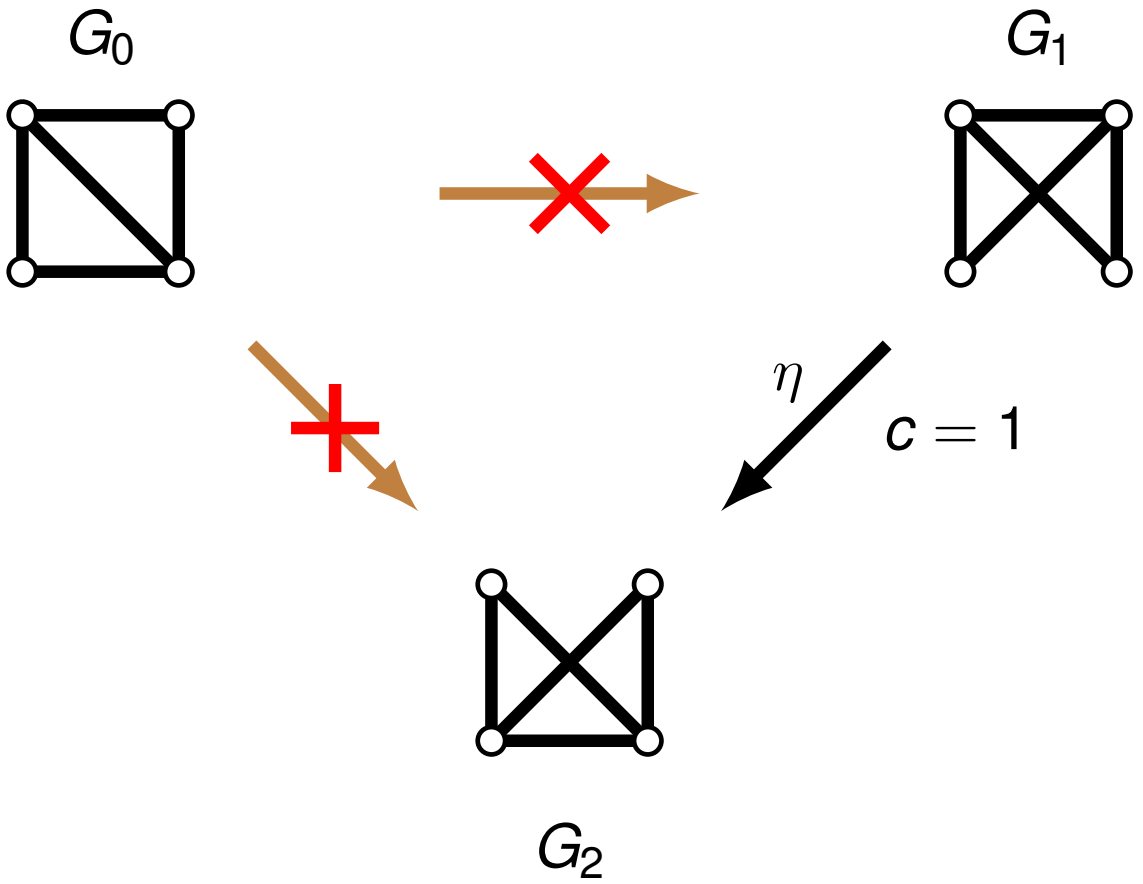
Soundness: ϕ is known from ψ_0 and ψ_1

Example



Simulation for $c = 0$: No need to know ψ_1 or ϕ

Example



Simulation for $c = 1$: No need to know ψ_0 or ϕ

Outline

General Description

P and V exchange 3 messages (a, c, t):

- P sends the commitment message a to V.
- V sends a random challenge c to P.
- P sends the response message t to V.

Knowledge extraction: From any two accepting transcripts (a, c_1, t_1), (a, c_2, t_2) with the same a but different c, the witness can be extracted.

Honest Verifier Zero-Knowledge: The honestly generated accepting transcript (a, c, t) can be simulated without knowing the witness.

Knowledge of a Discrete Logarithm

Prover	$\leftarrow (G, q, g, y = g^x) \rightarrow$	Verifier
$r \leftarrow \mathbb{Z}_q$; $a = g^r$; **send** a		**receive** a
receive c		$c \leftarrow \mathbb{Z}_q$; **send** c
$t = r + cx \bmod q$; **send** t		**receive** t
		accept if $g^t = ay^c$

Correctness: $g^t = g^{r+cx} = g^r(g^x)^c = ay^c$.

Extractability: (From (a, c_1, t_1) and (a, c_2, t_2))
$g^{t_2 - t_1} = y^{c_2 - c_1} \quad \Rightarrow \quad x = (t_2 - t_1)(c_2 - c_1)^{-1} \bmod q$.

Simulatability: Choose a random t, then compute $a = g^t y^{-c}$ for a random c.

Equality of two Discrete Logarithms

Prover		Verifier
	$\leftarrow (G, q, g, y = g^x) \rightarrow$	
	$\leftarrow (H, q, h, z = h^x) \rightarrow$	
$r \leftarrow \mathbb{Z}_q;\ a = g^r;\ b = h^r;$ **send**(a, b)		**receive**(a, b)
receive c		$c \leftarrow \mathbb{Z}_q;$ **send** c
$t = r + cx \bmod q;$ **send** t		**receive** t
		accept if $g^t = ay^c$
		and $h^t = bz^c$

Correctness: $g^t = g^{r+cx} = g^r(g^x)^c = ay^c$,
$h^t = h^{r+cx} = h^r(h^x)^c = bz^c$.

Extractability: $g^{t_2-t_1} = y^{c_2-c_1}$, $h^{t_2-t_1} = z^{c_2-c_1}$ $\Rightarrow$
$x = (t_2 - t_1)(c_2 - c_1)^{-1} \bmod q$.

Simulatability: Choose a random t, then compute $a = g^t y^{-c}$,
$b = h^t z^{-c}$ for a random c.

Knowledge of an e-th Root $\bmod n$

Prover	$\leftarrow (n = pq, e, y = x^e \bmod n) \rightarrow$	Verifier
$r \leftarrow \mathbb{Z}_n$; $a = r^e \bmod n$; **send** a		**receive** a
receive c		$c \leftarrow \mathbb{Z}_e$; **send** c
$t = rx^c \bmod n$; **send** t		**receive** t
		accept if $t^e = ay^c \bmod n$

Correctness: $t^e = (rx^c)^e = r^e(x^e)^c = ay^c \bmod n$.

Extractability: $(t_2/t_1)^e = y^{c_2-c_1} \quad \Rightarrow \quad x^{c_2-c_1} = t_2/t_1 \bmod n$.
For α, β s.t. $\alpha(c_2 - c_1) + \beta e = 1$, $x = (t_2/t_1)^\alpha y^\beta \bmod n$.

Simulatability: Choose a random t, then compute
$a = t^e y^{-c} \bmod n$ for a random c.

Knowledge of Representation

Prover	$\leftarrow (G, q, g, h, z = g^x h^y) \rightarrow$	Verifier
$r, s \leftarrow \mathbb{Z}_q$; $a = g^r h^s$; **send** a		**receive** a
receive c		$c \leftarrow \mathbb{Z}_q$; **send** c
$t = r + cx \bmod q$; $u = s + cy \bmod q$		
send(t, u)		**receive**(t, u)
		accept if $g^t h^u = az^c$

Correctness: $g^t h^u = g^{r+cx} h^{s+cy} = g^r h^s (g^x h^y)^c = az^c$.

Extractability: $g^{t_2 - t_1} h^{u_2 - u_1} = z^{c_2 - c_1} \quad \Rightarrow$
$x = (t_2 - t_1)(c_2 - c_1)^{-1} \bmod q$, $y = (u_2 - u_1)(c_2 - c_1)^{-1} \bmod q$
define a valid representation $z = g^x h^y$.

Simulatability: Choose random t, u, then compute
$a = g^t h^u z^{-c}$ for a random c.

Inequality of two Discrete Logarithms

Prover $\leftarrow (G, q, g, h, y = g^x, z \neq h^x) \rightarrow$ Verifier

---

$r, s \leftarrow \mathbb{Z}_q$; $a = y^r g^s$; $b = z^r h^s$
$v \leftarrow \mathbb{Z}_q^\times$; $w = z^v h^{-vx}$; **send**(a, b, w)
receive c
$t = r + cv \bmod q$; $u = s - cvx \bmod q$
send(t, u)

receive(a, b, w)
$c \leftarrow \mathbb{Z}_q$; **send** c

receive(t, u)
accept if $y^t g^u = a$ **and**
$z^t h^u = bw^c$ **and** $w \neq 1$

Correctness: $y^t g^u = g^{rx+cvx+s-cvx} = y^r g^s$,
$z^t h^u = z^r h^s (z^v h^{-vx})^c = bw^c$ and $z^v \neq h^{vx}$.

Extractability: $x = -(u_2 - u_1)(t_2 - t_1)^{-1} \bmod q$ since
$t_2 = t_1 \quad \Rightarrow \quad u_2 = u_1 \quad \Rightarrow \quad w = 1$.

Simulatability: Choose random t, u and $w \neq 1$, then compute $a = y^t g^u$ and $b = z^t h^u w^{-c}$ for a random c.

AND Proofs

Given two Sigma protocols Σ_1 and Σ_2 for two statements T_1 and T_2, we can build another Σ for the conjunction T_1 **and** T_2.

Prover	$\leftarrow (Pub_1, Pub_2) \rightarrow$	Verifier
Compute a_1 and a_2 as in Σ_1 and Σ_2		
send(a_1, a_2)		**receive**(a_1, a_2)
receive c		Select random c; **send** c
Compute t_1 and t_2 as in Σ_1 and Σ_2		
send(t_1, t_2)		**receive**(t_1, t_2)
		accept if $(a_1, c, t_1), (a_2, c, t_2)$ are accepted in Σ_1 and Σ_2

It requires that the honest distribution of c is the same in both Σ_1 and Σ_2.

Correctness, Extractability and ZK: Directly from Σ_1 and Σ_2.

OR Proofs

Now, for the disjunction T_1 **or** T_2:

Prover	$\leftarrow (Pub_1, Pub_2) \rightarrow$	Verifier

Assume it knows only the witness for T_1
Use Σ_2 simulator to obtain (a_2, c_2, t_2)
Compute a_1 as in Σ_1
send(a_1, a_2)
receive c
Compute t_1 for $c_1 = c - c_2$ as in Σ_1
send(t_1, t_2, c_1, c_2)

receive(a_1, a_2)
Select random c; **send** c

receive(t_1, t_2, c_1, c_2)
accept if $(a_1, c_1, t_1), (a_2, c_2, t_2)$ are accepted in Σ_1 and Σ_2, **and** $c = c_1 + c_2$

It requires the same set of challenges in Σ_1 and Σ_2, and with a group operation (e.g., addition).

Correctness, Extractability and ZK: Directly from Σ_1 and Σ_2.

Outline

Fiat-Shamir Transformation

From an interactive Sigma protocol Σ and a hash function $H(\cdot)$ we define a non-interactive proof:

Prover	$\leftarrow (Pub) \rightarrow$	Verifier
Compute a as in Σ $c = H(Pub, a)$ Compute t as in Σ **send**(a, c, t)		**receive**(a, c, t) **accept if** $c = H(Pub, a)$ **and** (a, c, t) is accepted in Σ

In the Random Oracle Model, c is uniformly distributed, as if it was generated by the honest V.

Fiat-Shamir Transformation

Properties:

- Correctness is straight forward from Σ.
- The simulation of the Random Oracle $H(\cdot)$ provides the Extractability and ZK properties, using the same extractor and simulator of Σ.
- The proof can be compressed to (a, t), since V can compute $c = H(Pub, a)$.

Example: Non-Interactive proof of knowledge of x s.t. $y = g^x$:

Proof: $\pi = (a = g^r, t = r + xH(G, q, g, y, a))$.

Verification: $g^t = ay^{H(G,q,g,y,a)}$.

Outline

Statements Related to ElGamal Encryption

- **Knowledge of the secret key:** Given the public key $y = g^x$, prove the knowledge of the discrete logarithm x.
- **Knowledge of decryption by the sender:** Given $pk = g^x$ and a ciphertext $(c_1, c_2) = (g^r, y^r m)$, prove the knowledge of r as the discrete logarithm of c_1.
- **Knowledge of decryption by the recipient:** Given $pk = g^x$ and a ciphertext $(c_1, c_2) = (g^r, y^r m)$, prove the knowledge of x as the discrete logarithm of y.

Statements Related to ElGamal Encryption

- **Correct encryption:** Given $pk = g^x$, a ciphertext $(c_1, c_2) = (g^r, y^r m)$ and m, prove the equality of the discrete logarithms of c_1 and c_2/m with respect to the bases g and y.
- **Correct decryption:** Given $pk = g^x$, a ciphertext $(c_1, c_2) = (g^r, y^r m)$ and m, prove the equality of the discrete logarithms of y and c_2/m with respect to the bases g and c_1.
- **Equality of plaintexts:** Given $pk = g^x$ and the ciphertexts (c_1, c_2) and (c'_1, c'_2), prove the equality of the discrete logarithms of y and c'_2/c_2 with respect to the bases g and c'_1/c_1.

Statements Related to ElGamal Encryption

- **Strong Homomorphic evaluation:** Given $pk = g^x$ and ciphertexts $(c_1, c_2) = (g^r, y^r m)$, $(c_1', c_2') = (g^{r'}, y^{r'} m')$ and $(c_1'', c_2'') = (g^{r''}, y^{r''} m'')$, prove that $m'' = mm'$ by proving the equality of the discrete logarithms of $c_1''/(c_1 c_1')$ and $c_2''/(c_2 c_2')$ with respect to the bases g and y. The witness is $r'' - r - r'$.
- **Anonymous Knowledge of decryption:** Use an OR proof that you know either the randomness r in a ciphertext, or the secret key x.
- **Encrypted ballot validity:** Prove that a ciphertext encrypts either 1 or g with an OR proof of knowledge of the decryption.

Deniable (Directed) Signature

Generation of a signature that only convinces a specific verifier and not anyone else:

- Use Fiat-Shamir to produce an non-interactive OR proof of the statement "I know sk_A OR I know sk_B, and endorse message m".
- The "endorse the message m" part is fulfilled by including m in the hash of the non-interactive proof.
- If B did not generated the signature, he knows that it had to be generated by A.
- B cannot convince any C that the signature was generated by A, since it can be generated by himself.

Data Protection

Jorge L. Villar

MCYBERS, UPC, Fall 2025

END