Secure computing, sixth lecture

Zero-knowledge proofs

Xavier Leroy

2025-12-11

Collège de France, chair of Software sciences
`xavier.leroy@college-de-france.fr`

Zero-knowledge proofs (ZKP)

At the start of the protocol:

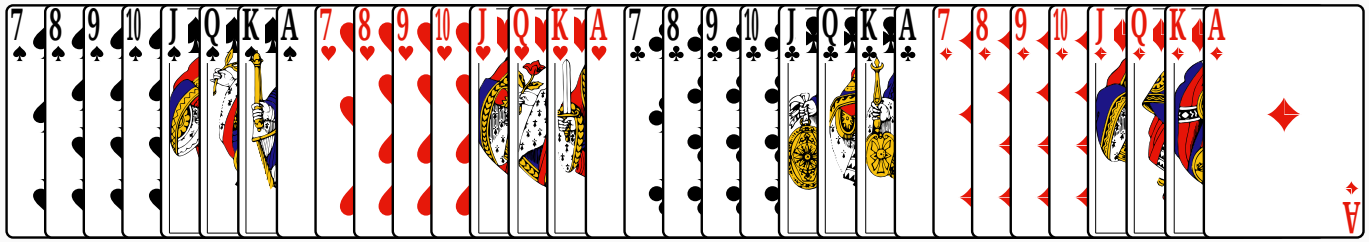
- The prover knows a secret x that satisfies a property $P(x)$.

At the end of the procol:

- The verifier is convinced that the prover knows x s.t. $P(x)$.
- The verifier learned nothing about x that is not implied by $P(x)$.

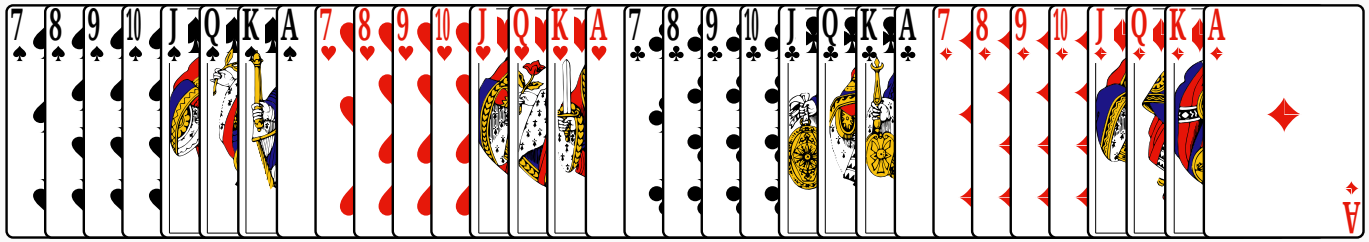
An example of a zero-knowledge proof (reminder)

Peggy (the prover) draws a card from a deck and wishes to convince Victor (the verifier) that it is a red card, without showing him the card.



An example of a zero-knowledge proof (reminder)

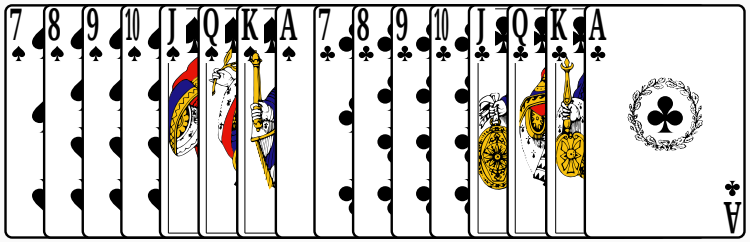
Peggy (the prover) draws a card from a deck and wishes to convince Victor (the verifier) that it is a red card, without showing him the card.

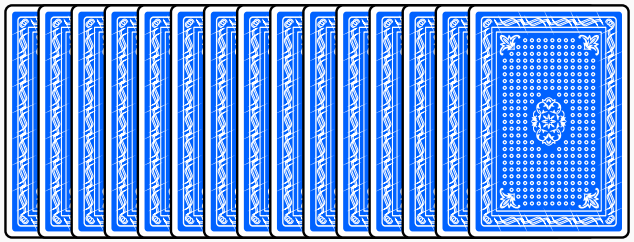


1. Peggy and Victor check the deck of cards:
16 red cards, 16 black cards.

An example of a zero-knowledge proof (reminder)

Peggy (the prover) draws a card from a deck and wishes to convince Victor (the verifier) that it is a red card, without showing him the card.

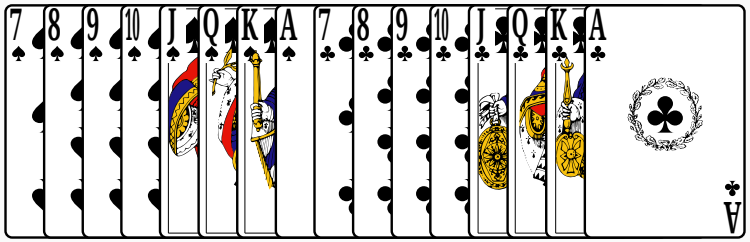


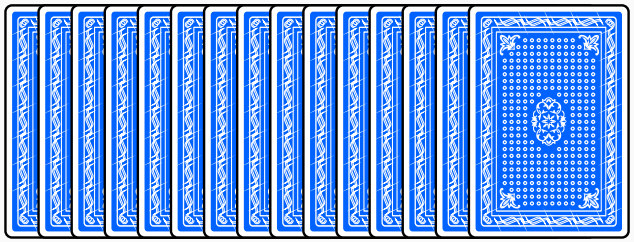


2. Peggy takes the cards, keeps one, puts the other cards face down, and turns 16 black cards over.

An example of a zero-knowledge proof (reminder)

Peggy (the prover) draws a card from a deck and wishes to convince Victor (the verifier) that it is a red card, without showing him the card.





3. Victor is convinced that the card kept by Peggy is red, but knowns nothing else about this card.

Some uses of zero-knowledge proofs

Proving facts about private data

(“my vote is well formed”, “there’s enough money in my account”, “I am 18 or over”, etc)

Anonymous authorization

(accessing non-free Web sites; anonymity in cryptocurrencies)

Delegating computations with correctness guarantees

(“rollups” on blockchains)

Sigma protocols

The shape of a Sigma protocol

Peggy must convince Victor that she knows x such that $P(x)$.

A 3-exchange protocol: commitment, challenge, response.

	Prover		Verifier
Commitment	$k \leftarrow \mathit{Commit}(x)$	$\xrightarrow{k}$	
Challenge		$\xleftarrow{c}$	$c \leftarrow \mathit{Challenge}$
Response	$r \leftarrow \mathit{Resp}(x, c)$	$\xrightarrow{r}$	
Verification			$\mathit{Verif}(k, c, r)$

Example of a Sigma protocol: Schnorr's protocol (knowledge of a discrete logarithm)

Peggy must convince Victor that she knows x such that $g^x = a$ (g generator of a group G of order q; $a \in G$, known to all).

	Prover		Verifier
Commitment	$y \in \mathbb{Z}_q$ random		
	$k = g^y$	$\xrightarrow{k}$	
Challenge		$\xleftarrow{c}$	$c \in \mathbb{Z}_q$ random
Response	$r = y + x \cdot c \pmod q$	$\xrightarrow{r}$	
Verification			$g^r = k \cdot a^c \ ?$

(Application: Peggy can authenticate herself without revealing her password x.)

Expected properties of a zero-knowledge proof

Completeness: if both participants follow the protocol and if the prover knows x such that $P(x)$, the verification always succeeds.

Correctness: if the verification succeeds, the prover knows x such that $P(x)$ with overwhelming probability.

Zero-knowledge: the verifier learns nothing about x beyond what it can deduce from the fact $P(x)$.

Completeness of a Sigma protocol

Completeness: if both participants follow the protocol and if the prover knows x such that $P(x)$, the verification always succeeds.

$$\begin{aligned} k &\leftarrow \mathit{Commit}(x) \\ c &\leftarrow \mathit{Challenge} \\ r &\leftarrow \mathit{Resp}(x, c) \end{aligned} \quad \Longrightarrow \quad \mathit{Verif}(k, c, r) \text{ succeeds}$$

In the case of Schnorr's protocol:

$$\begin{aligned} k &= g^y \\ r &= y + x \cdot c \end{aligned} \quad \Longrightarrow \quad g^r = g^y \cdot (g^x)^c = k \cdot a^c$$

Correctness of a Sigma protocol

Correctness: if verification succeeds, the prover knows x such that $P(x)$ with overwhelming probability.

Idea: if the prover doesn't reply randomly, and if we have full access to the code of the prover and to the values of its local variables, we can recover x.

A recovery method that works for most Sigma protocols: execute the protocol twice while forcing the prover to commit on the same k twice.

Special correctness of a Sigma protocol

Special correctness: there exists an extraction function $Extr$ in polynomial probabilistic time that recovers x from two valid protocol traces (k, c_1, r_1) and (k, c_2, r_2) having the same commitment k and different challenges $c_1 \neq c_2$.

$$x = Extr((k, c_1, r_1), (k, c_2, r_2))$$
$$\text{satisfies } P(x) \text{ with overwhelming probability}$$

In the case of Schnorr's protocol:

$$\begin{aligned} r_1 &= y + x \cdot c_1 \\ r_2 &= y + x \cdot c_2 \end{aligned} \quad \Longrightarrow \quad x = \frac{r_1 - r_2}{c_1 - c_2} \pmod{q}$$

Zero-knowledge for a Sigma protocol

Zero-knowledge: the verifier learns nothing about x beyond what he can deduce from the fact $P(x)$.

Idea: if the prover did not know x but could time-travel and change her commitment k after receiving the challenge c, would she be able to produce a correct reply r?

If so, we see that the existence of a trace (k, c, r) that passes verification implies nothing about x, not even that it exists.

Zero-knowledge for a Sigma protocol

Zero-knowledge: there exists a function $Simu(c) = (k, r)$ producing a valid trace (k, c, r) for a given challenge c.

$$\begin{array}{l} c \leftarrow Challenge \\ (k, r) \leftarrow Simu(c) \end{array} \implies Verif(k, c, r) \text{ succeeds}$$

In the case of Schnorr's protocol, take

$$Simu(c) = (g^r \cdot a^{-c}, r) \quad \text{for a random } r \in \mathbb{Z}_q$$

The verification $g^r = k \cdot a^c$ passes, since $k \cdot a^c = g^r \cdot a^{-c} \cdot a^c = g^r$.

Generalization: Maurer's protocol

(U. Maurer, *Unifying Zero-Knowledge Proofs of Knowledge*, AFRICACRYPT 2009.)

Two groups G, H of order q and a one-way morphism $\varphi : G \to H$.

Given $a \in H$, prove that $\exists x \in G,\ \varphi(x) = a$.

	Prover		Verifier
Commitment	$y \in G$ random		
	$k = \varphi(y)$	$\xrightarrow{k}$	
Challenge		$\xleftarrow{c}$	$c \in \mathbb{Z}_q$ random
Response	$r = y \cdot x^c$	$\xrightarrow{r}$	
Verification			$\varphi(r) = k \cdot a^c$?

Security of Maurer's protocol

Completeness: if $a = \varphi(x)$ and $k = \varphi(y)$ and $r = y \cdot x^c$,

$$\varphi(r) = \varphi(y) \cdot \varphi(x)^c = k \cdot a^c \quad \text{(since } \varphi \text{ is a morphism)}$$

Correctness: given two executions $r_1 = y \cdot x^{c_1}$ and $r_2 = y \cdot x^{c_2}$, we have $r_1/r_2 = x^{c_1}/x^{c_2} = x^{c_1 - c_2}$ hence $x = (r_1/r_2)^{c_2 - c_1}$.

Zero-knowledge: for a given c, we simulate a valid execution (k, c, r) by choosing $r \in G$ randomly and taking $k = \varphi(r) \cdot a^{-c}$.

We have $k \cdot a^r = \varphi(r) \cdot a^{-c} \cdot a^r = \varphi(r)$.

Instantiations of Maurer's protocol

Schnorr: knowledge of a discrete logarithm $\exists x,\ g^x = a$

$\varphi : \mathbb{Z}_q \to G \qquad \varphi(x) = g^x$

Guillou-Quisquater: knowledge of a e-th root $\exists x,\ x^e = a \bmod m$

$\varphi : \mathbb{Z}_m^* \to \mathbb{Z}_m^* \qquad \varphi(x) = x^e \bmod m$

Chaum-Pedersen: $\exists x,\ a = g^x \wedge b = h^x$ that is, $\log_g(a) = \log_h(b)$

$\varphi : \mathbb{Z}_q \to G \times G \qquad \varphi(x) = (g^x, h^x)$

(a, b) **is an ElGamal encryption of** m**:** $\exists x,\ a = g^x \wedge b = h^x \cdot m$

(Example used in lecture #1. Use Chaum-Pedersen with a and b/m.)

Prouver une disjonction

Étant donnés des protocoles Sigma pour les propriétés P_1 et P_2,
peut-on construire une preuve Sigma pour la disjonction $P_1 \vee P_2$?

Exemple d'utilisation: prouver qu'un bulletin de vote chiffré avec ElGamal est soit un chiffré de 0 soit un chiffré de 1,
sans révéler dans quel cas on se trouve.

Idée: une réponse à un défi est une paire de réponses:
une "vraie" réponse construite avec $Resp_i$ (si P_i est vraie);
une réponse simulée construite avec $Simu_j$ pour $j \neq i$.

Zero-knowledge proof of a disjunction $P_1 \vee P_2$

Commitment: a pair (k_1, k_2) of commitments for the protocols P_1 and P_2.

Response: a quadruple (c_1, c_2, r_1, r_2)
where r_1 is a P_1-response to challenge c_1
and r_2 a P_2-response to challenge c_2.

Verification of a response to the challenge c:

$$\begin{aligned} \mathit{Verif}(\ (k_1, k_2),\ c,\ (c_1, c_2, r_1, r_2)\ ) = \quad & c = c_1 \oplus c_2 \\ & \wedge\ \mathit{Verif}_1(k_1, c_1, r_1) \\ & \wedge\ \mathit{Verif}_2(k_2, c_2, r_2) \end{aligned}$$

The verification is symmetrical in P_1 and P_2, and gives no hint to which property P_1 or P_2 is true.

Zero-knowledge proof of a disjunction $P_1 \vee P_2$

Assume $P_1(x)$ is true. In general, we have no proof for $P_2(x)$, so the prover simulates one for a challenge c_2 of her choosing.

	Prover		Verifier
Commitment	$k_1 \leftarrow Commit_1(x)$		
	$c_2 \leftarrow Challenge$		
	$(k_2, r_2) \leftarrow Simu_2(c_2)$	$\xrightarrow{k_1, k_2}$	
Challenge		$\xleftarrow{c}$	$c \leftarrow Challenge$
Response	$c_1 = c \oplus c_2$		
	$r_1 \leftarrow Resp_1(x, c_1)$	$\xrightarrow{c_1, c_2, r_1, r_2}$	
Verification			$c = c_1 \oplus c_2$?
			$Verif_1(k_1, c_1, r_1)$?
			$Verif_2(k_2, c_2, r_2)$?

Non-interactive proofs

Towards a non-interactive protocol

	Prover		Verifier
Commitment	$k \leftarrow Commit(x)$	$\xrightarrow{k}$	
Challenge		$\xleftarrow{c}$	$c \leftarrow Challenge$
Response	$r \leftarrow Resp(x, c)$	$\xrightarrow{r}$	
Verification			$Verif(k, c, r)$

The challenge c does not need to be chosen by the verifier. Anyone could choose c, provided that

- c is chosen after the commitment;
- c is not controlled by the prover;
- c is "random enough".

The Fiat-Shamir heuristic

(A. Fiat, A. Shamir, *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*, CRYPTO 1986.)

Use a hash function $\mathcal{H}$ to produce the challenge from the commitment k and the public parameters pp of the problem:

$$c = \mathcal{H}(pp \parallel k)$$

If $\mathcal{H}$ is viewed as a random oracle, c is random; moreover, we cannot manipulate k to obtain a c of our choosing.

If $\mathcal{H}$ is a standard cryptographic hash function (such as SHA-256), we strongly believe these properties of c still hold.

Making a Sigma protocol non-interactive

	Prover		Verifier
Commitment	$k \leftarrow \mathit{Commit}(x)$		
Challenge	$c = \mathcal{H}(pp \parallel k)$		
Response	$r \leftarrow \mathit{Resp}(x, c)$	$\xrightarrow{k,r}$	
Verification			$c = \mathcal{H}(pp \parallel k)$
			$\mathit{Verif}(k, c, r)$

Fiat-Shamir: the challenge is obtained from k and pp using $\mathcal{H}$.

The proof is the pair (k, r). It can be checked at any time and as many times as desired, without interacting with the prover.

Example of a non-interactive proof

I know the discrete logarithm of a in base g in $\mathbb{Z}_p^*$.

$$p = 25644269200652980450766820164246153935 3$$

$$g = 781944113$$

$$a = 660237491474363027736483369857459075 35$$

Here is my proof:

$$k = 200299568312215464498549432374020738 31$$

$$r = 221824598860809771154727139215467720 68$$

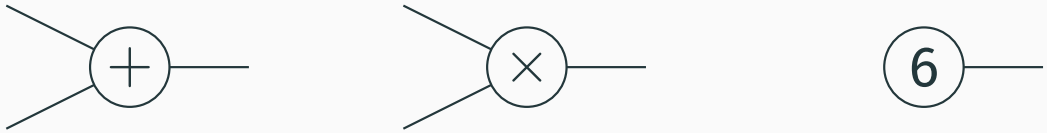
(Schnorr protocol + Fiat-Shamir with SHA-256 hash).

Arithmetic circuits and quadratic programs

Arithmetic circuits

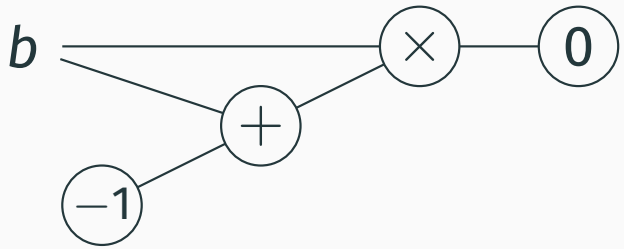
Combinatorial circuits that go beyond standard Boolean circuits:

- Wires carry values taken from a field $\mathbb{F}_q$ of order q.
- Base gates: addition, multiplication, constants.



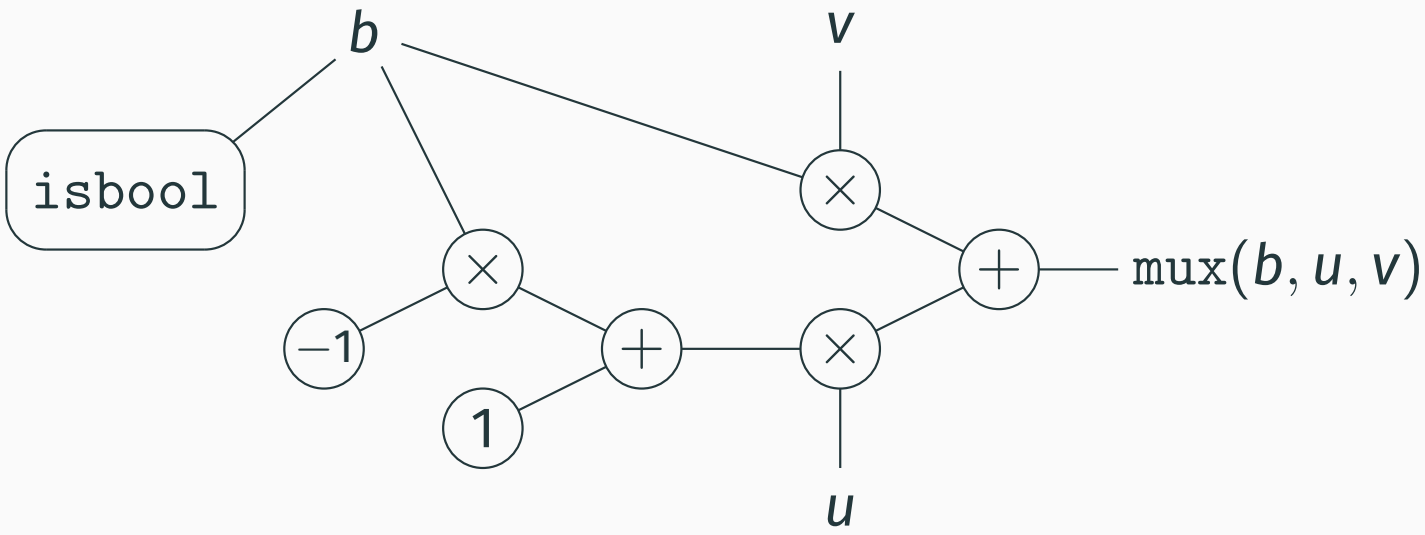
- We can constrain the output of a gate to be equal to a constant or to the output of another gate.

  Example: the $\texttt{isbool}(b)$ circuit that enforces $b \in \{0, 1\}$.



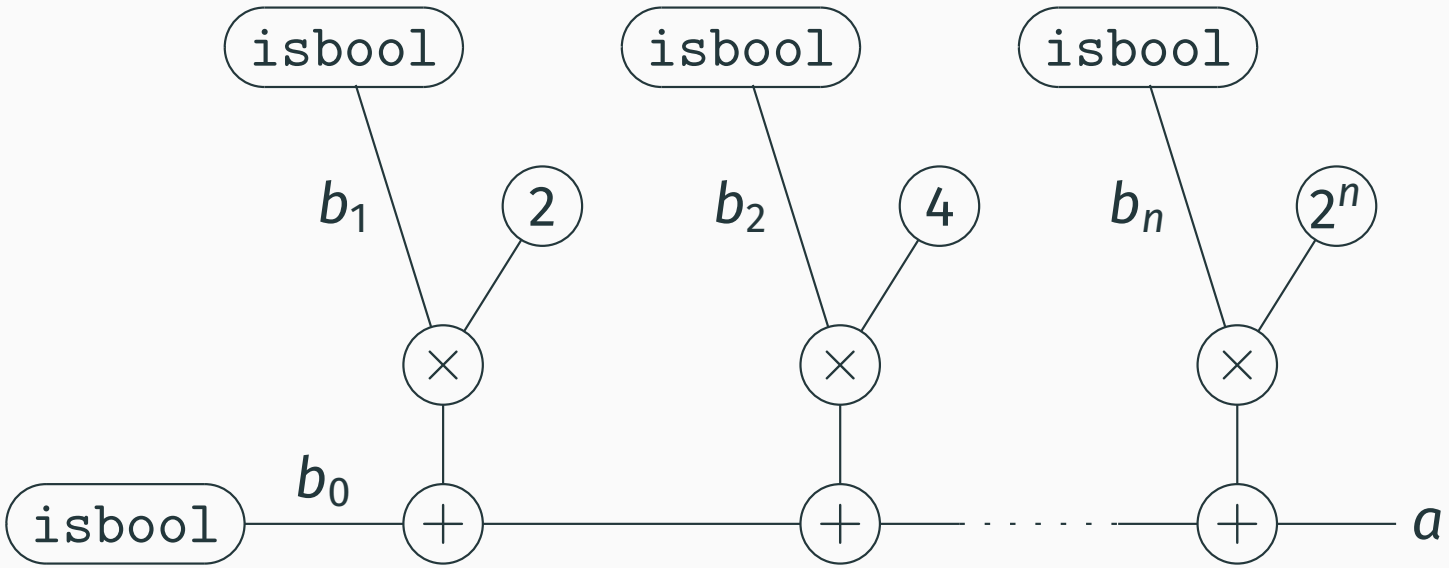
Examples of arithmetic circuits

Multiplexer: $\texttt{mux}(b, u, v) = \texttt{if } b \texttt{ then } v \texttt{ else } u$



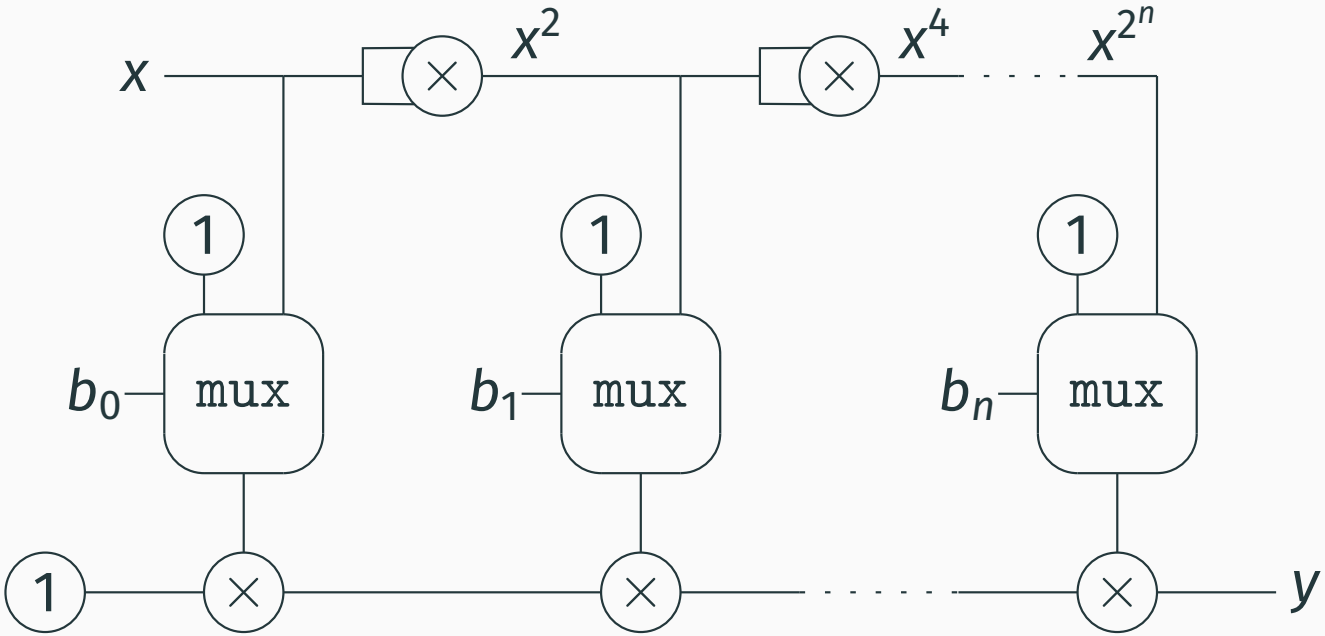
Combining arithmetic operations and Boolean operations

Binary decomposition: $a = b_0 + 2b_1 + \ldots + 2^n b_n$



Combining arithmetic operations and Boolean operations

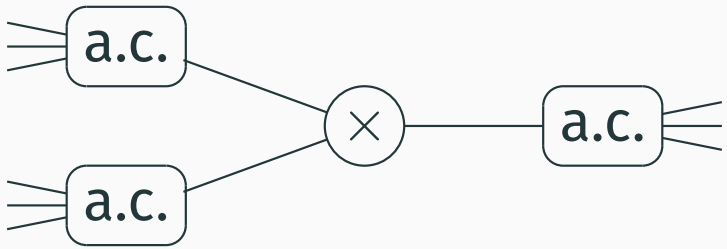
Fast exponentiation: $y = x^a$ where $a = b_0 + 2b_1 + \cdots 2^n b_n$.



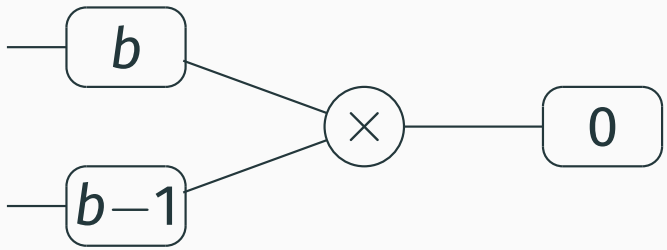
If we constrain x and y, this circuit proves the knowledge of the discrete logarithm of y in base x .

From gates to equations

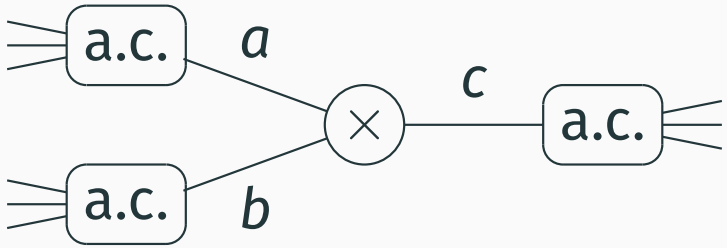
We decompose the circuit in macro-gates comprising exactly one “product” gate each, with two inputs and one output that are affine combinations of wires.



In the case of the `isbool` circuit:



From gates to equations



Let $w_1, \ldots, w_n$ be the values of the n wires of the circuit. Take $w_0 = 1$.

The two inputs a, b and the output c of the product gate are linear combinations of the w_i:

$$a = a_0 w_0 + \cdots + a_n w_n \quad b = b_0 w_0 + \cdots + b_n w_n \quad c = c_0 w_0 + \cdots + c_n w_n$$

Since $c = a \times b$, we obtain the constraint

$$(a_0 w_0 + \cdots + a_n w_n)\,(b_0 w_0 + \cdots + b_n w_n) = c_0 w_0 + \cdots + c_n w_n$$

Rank-1 constraint system

Applying this encoding to each of the m product gates of the original circuit, we obtain a set of m equations of degree 2 with unknowns w_i.

$$\begin{aligned}
(a_{1,0}w_0 + \cdots + a_{1,n}w_n)\,(b_{1,0}w_0 + \cdots + b_{1,n}w_n) &= c_{1,0}w_0 + \cdots c_{1,n}w_n \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
(a_{m,0}w_0 + \cdots + a_{m,n}w_n)\,(b_{m,0}w_0 + \cdots + b_{m,n}w_n) &= c_{m,0}w_0 + \cdots c_{m,n}w_n
\end{aligned}$$

The coefficients $a_{j,i}$, $b_{j,i}$, $c_{j,i}$ describe the structure of the circuit.

Quadratic Arithmetic Program

An encoding of a rank-1 constraint system using polynomials.

We choose m points $x_1, \ldots, x_m$ of $\mathbb{F}_q$.
The point x_j identifies the j-th product gate.

Using Lagrange interpolation, we build polynomials $A_0, \ldots, A_n$, $B_0, \ldots, B_n$, $C_0, \ldots, C_n$ such that

$$A_i(x_j) = a_{j,i} \qquad B_i(x_j) = b_{j,i} \qquad C_i(x_j) = c_{j,i}$$

Then, we build the polynomial

$$P = (w_0 A_0 + \cdots + w_n A_n)\,(w_0 B_0 + \cdots + w_n B_n) - (w_0 C_0 + \cdots + w_n C_n)$$

Quadratic Arithmetic Program

$$P = (w_0 A_0 + \cdots + w_n A_n)\,(w_0 B_0 + \cdots + w_n B_n) - (w_0 C_0 + \cdots + w_n C_n)$$

The initial circuit executes correctly if and only if

$$P(x_j) = 0 \text{ for each } j \in \{1, \ldots, m\}$$

or, equivalently, if and only if P can be written as

$$P = H\,T \quad \text{where} \quad T = (X - x_1)\,(X - x_2) \cdots (X - x_m)$$

Connection with zero-knowledge proofs

Our proofs are “arguments of knowledge”:

I know values $\mathbf{x}$ such that $\mathit{Prop}(\mathbf{x})$ holds.

If the property *Prop* can be expressed as a circuit C, the argument of knowledge becomes

I know wire values $\mathbf{w}$ that satisfy the circuit C.

After encoding the circuit as a QAP, the argument becomes

I know values $\mathbf{w}$ and a polynomial H such that

$$(w_0 A_0 + \cdots + w_n A_n)\,(w_0 B_0 + \cdots + w_n B_n) - (w_0 C_0 + \cdots + w_n C_n) = H\,T$$

where the polynomials A_i, B_i, C_i and the target polynomial T are publically known and depend only on the circuit.

Masked evaluation of polynomials

Proving equality of two polynomials

Alice knows a degree-n polynomial $A = a_0 + a_1 X + \cdots + a_n X^n$.

Bob knows a degree-n polynomial $B = b_0 + b_1 X + \cdots + b_n X^n$.

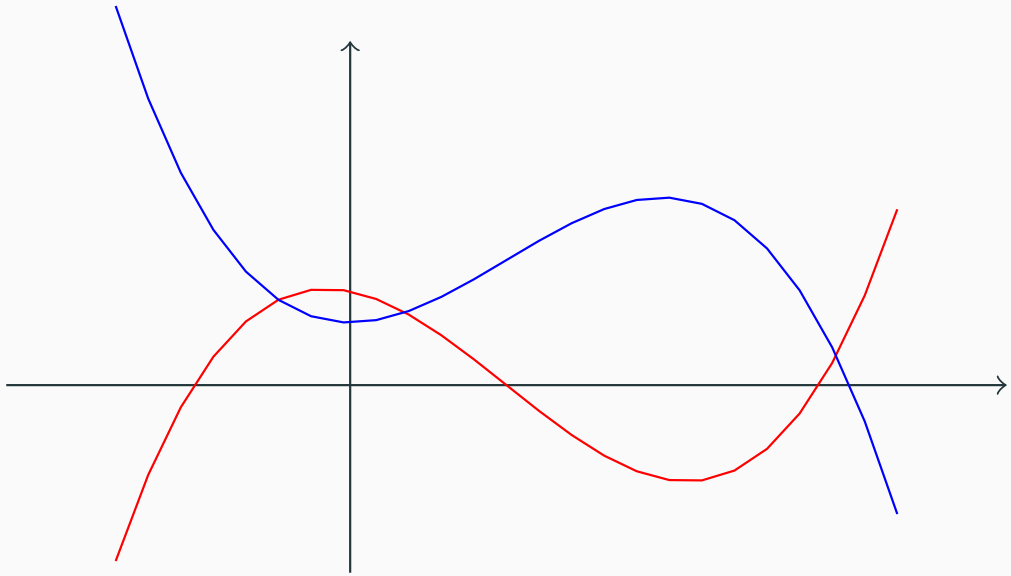
They want to check that $A = B$.

Naive proof: check that $a_0 = b_0, \ldots, a_n = b_n$
$\Rightarrow$ proof of size $\mathcal{O}(n)$.

Succint proof: evaluate A and B at a random point z.
If $A(z) = B(z)$, then $A = B$ with high probability.

Proving equality of two polynomials

If two degree-n polynomials A and B are different, they coincide on at most n points.



We use a finite field $\mathbb{F}_q$ of order $q \gg n$. The probability that $z \in \mathbb{F}_q$ is one of the intersection points is $n/q \ll 1$.

Hence, if $A(z) = B(z)$, we have $A = B$ with high probability $1 - n/q$.

Proving that a polynomial has some given roots (first try)

Peggy has a secret polynomial P and wants to convince Victor that P is zero at points $x_1, \ldots x_n$.

Prover		Verifier
		$T = (X - x_1) \cdots (X - x_n)$
	$\xleftarrow{T,z}$	$z \in \mathbb{F}_q$ random
$H = P/T$		
$p = P(z) \quad h = H(z)$	$\xrightarrow{p,h}$	
		$p = h \cdot T(z)$?

If $p = h \cdot T(z)$, with high probability we have $P = H \cdot T$ and therefore $x_1, \ldots, x_n$ are roots of P.

Masking the evaluation of a polynomial

In the previous protocol, the prover can easily cheat.
For example, she picks h at random and takes $p = h \cdot T(z)$.

Idea: to limit what the prover can compute, we transmit z encrypted and we use homomorphic evaluation.

Let E be a one-way function that is homomorphic for addition and multiplication by a constant:

$$E(x + y) = E(x) \oplus E(y)$$
$$E(c \cdot x) = c \odot E(x)$$

but definitely not homomorphic for general multiplication:

$E(x \cdot y)$ cannot be computed easily from $E(x)$ and $E(y)$

Example of homomorphic one-way function

Let $(G, \cdot)$ be a group of order q where discrete logarithms are hard to compute. Let g be a generator of G.

Define $E : \mathbb{F}_q \to G$ as $E(x) = g^x$.

We have

$$E(x + y) = g^{x+y} = g^x \cdot g^y = E(x) \cdot E(y) \qquad (\oplus \text{ is the product in } G)$$
$$E(c \cdot x) = g^{cx} = (g^x)^c = E(x)^c \qquad (\odot \text{ is exponentiation in } G)$$

But from g^x and g^y we cannot easily compute g^{xy} (Diffie-Hellman hypothesis).

Note: equality of ciphertexts $E(x) = E(y)$ implies equality of plaintexts $x = y \pmod{q}$.

Homorphic evaluation of a polynomial

We can evaluate linear combinations homomorphically:

$$E(c_1 x_1 + \cdots + c_n x_n) = c_1 \odot E(x_1) \oplus \cdots \oplus c_n \odot E(x_n)$$

If we are given the encryptions of the first n powers of z

$$E(1), E(z), \ldots, E(z^n)$$

we can evaluate $P(z)$ for any polynomial P of degree n:

$$E(c_0 + c_1 z + \cdots + c_n z^n) = c_0 \odot E(1) \oplus c_1 \odot E(z) \oplus \cdots \oplus c_n \odot E(z^n)$$

Proving that a polynomial has some given roots (second try)

Prover		Verifier
		$T = (X - x_1) \cdots (X - x_n)$
		$z \in \mathbb{F}_q$ random
	$\xleftarrow{T, z_0, \ldots, z_n}$	$z_i = E(z^i)$ for $i = 0, \ldots, n$
$H = P/T$		
$p = E(P(z))$ $\quad h = E(H(z))$	$\xrightarrow{p, h}$	
		$p = T(z) \odot h$?

Proving that a polynomial has some given roots (second try)

Prover		Verifier
		$T = (X - x_1) \cdots (X - x_n)$
		$z \in \mathbb{F}_q$ random
	$\xleftarrow{T, z_0, \ldots, z_n}$	$z_i = E(z^i)$ for $i = 0, \ldots, n$
$H = P/T$		
$p = E(P(z))$ $\quad h = E(H(z))$	$\xrightarrow{p, h}$	
		$p = T(z) \odot h$?

Note: the prover can still cheat!

E.g. $h = E(r)$ and $p = r \odot E(T(z))$ for a random r.

Authenticated encryption

Let $k \in \mathbb{F}_q$ be a secret known only to the verifier.

An authenticated encryption of x is a pair of encryptions of x and kx:

$$\bar{E}(x) = (E(x), E(kx))$$

Knowing k, the verifier can easily check that the encryption (a, b) is valid by testing $b = k \odot a$.

Authenticated encryption

$$\bar{E}(x) = (E(x), E(kx))$$

Without knowing k, the prover can compute homomorphically on authenticated ciphertexts:

$$\bar{E}(x + y) = \bar{E}(x) \oplus \bar{E}(y) \qquad \text{(pointwise)}$$
$$\bar{E}(c.\, x) = c \odot \bar{E}(x) \qquad \text{(pointwise)}$$

Without knowing k, the only valid authenticated ciphertexts that the prover can construct are linear combinations of authenticated ciphertexts provided by the verifier.

(This is known as the Knowledge of Exponent Assumption (KEA).)

Proving that a polynomial has some given roots (secure version)

Prover		Verifier
		$T = (X - x_1) \cdots (X - x_n)$
		$z \in \mathbb{F}_q$ random
	$\xleftarrow{T, z_0, \ldots, z_n}$	$z_i = \bar{E}(z^i)$ for $i = 0, \ldots, n$
$H = P/T$		
$p = \bar{E}(P(z))$ $\quad h = \bar{E}(H(z))$	$\xrightarrow{p, h}$	
		check validity of p and h
		$p = T(z) \odot h$?

Zero-knowledge version

To make the protocol zero-knowledge, it is enough to mask the responses of the prover by a random multiplicative factor r:

$$p = r \odot \bar{E}(P(z)) \qquad h = r \odot \bar{E}(H(z))$$

The verification $p = T(z) \odot h$ still works.

Non-interactive version

To obtain a non-interactive protocol, we need to choose the secrets z and k in advance, and to share encrypted data with the provers and the verifiers:

Evaluation key: $\bar{E}(1), \bar{E}(z), \ldots, \bar{E}(z^n)$

Verification key: $E(1), E(T(z)), E(k)$

Problem: without knowing k in the clear, how can we check the validity of an authenticated ciphertext (a, b) ?
(It is no longer possible to test $b = k \odot a$.)

$\Rightarrow$ Use a pairing.

Pairing (bilinear map)

Given two multiplicative groups G (generated by g) and G', a pairing $e : G \times G \to G'$ is a non-degenerate bilinear map:

$$e(g^a, g^b) = e(g, g^b)^a = e(g^a, g)^b = e(g, g)^{ab}$$

$$e(g, g) \neq 1 \text{ is a generator of } G'$$

Efficiently-computable pairings exist for some elliptic curves.

A pairing gives a way to homomorphically check the equality of two products:

$$e(g^a, g^b) = e(g^c, g^d) \iff ab = cd$$

Pairing-based verifications

Validity of an authenticated ciphertext (a, b):

Knowing k: $b = k \odot a$

Knowing only $E(k)$: $e(b, E(1)) = e(E(k), a)$

Verification of $P(z) = H(z) \cdot T(z)$:

Knowing $T(z)$: $p = T(z) \odot h$

Knowing only $E(T(z))$: $e(p, E(1)) = e(E(T(z)), h)$

The Pinocchio protocol: ZK-SNARKs for QAP

The Pinocchio protocol

(B. Parno, J. Howell, C. Gentry, M. Raykova, *Pinocchio: Nearly Practical Verifiable Computation*, S&P 2013, CACM 2016.)

A ZK-SNARK protocol:

Zero-Knowledge	
Succint	(proofs of constant size)
Non-interactive	
ARgument of Knowledge	(there exists $\mathbf{s}$ such that $Prop(\mathbf{s})$)

The property *Prop* is expressed as an arithmetic circuit represented in QAP form as degree-m polynomials

$$T, A_0, \ldots, A_n, B_0, \ldots, B_n, C_0, \ldots, C_n$$

where n is the number of secrets and m the number of product gates in the circuit.

Argument of knowledge

The prover must demonstrate that it knows secrets $s_0, \ldots, s_n$ and a poynomial H such that

$$(s_0 A_0 + \cdots + s_n A_n)\,(s_0 B_0 + \cdots + s_n B_n) - (s_0 C_0 + \cdots + s_n C_n) = H\,T$$

We write $\quad A = \sum s_i A_i \quad B = \sum s_i B_i \quad C = \sum s_i C_i\,.$

We use a variant of the protocol that proves that a polynomial has some given roots:

- Show that $A(z)\,B(z) - C(z) = H(z)\,T(z)$ for a random z.
- Moreover, show that A, B, C are linear combinations of the A_i, B_i, C_i with the same coefficients $s_0, \ldots, s_n$.

Protocol setup

An authority picks random $z, k, \alpha, \beta, \gamma, \delta$ in $\mathbb{F}_q$ and publishes the following "Common Reference String" (CRS), to be used by all provers and verifiers:

Evaluation key:

$\bar{E}(z^i)$ for $i = 0, \ldots, m$

$\bar{E}(A_i(z))$ for $i = 0, \ldots, n$

$\bar{E}(B_i(z))$ for $i = 0, \ldots, n$

$\bar{E}(C_i(z))$ for $i = 0, \ldots, n$

$E(\alpha A_i(z) + \beta B_i(z) + \gamma C_i(z))$ for $i = 0, \ldots, n$

Verification key:

$E(1)$ $E(k)$ $E(T(z))$

$E(\delta)$ $E(\alpha\delta)$ $E(\beta\delta)$ $E(\gamma\delta)$

Proof generation

Knowing the secrets $s_0, \ldots, s_n$, the prover

- builds the polynomials $A = \sum s_i A_i \quad B = \sum s_i B_i \quad C = \sum s_i C_i$
- computes H such that $A\ B - C = H\ T$ by polynomial division.

The proof consists of the authenticated encryptions of the values of A, B, C, H at point z:

$$\bar{E}(A(z)) \qquad \bar{E}(B(z)) \qquad \bar{E}(C(z)) \qquad \bar{E}(H(z))$$

and of the "checksum"

$$E(\alpha A(z) + \beta B(z) + \gamma C(z)) = \bigoplus s_i \odot E(\alpha A_i(z) + \beta B_i(z) + \gamma C_i(z))$$

(Size of the proof: 9 elements of group G, independently of the size m of the circuit and the number n of secrets.)

(Zero-knowledge: add a random multiple of T to A, B, C.)

Proof verification

The verifier receives four authenticated ciphertexts a, b, c and h, plus an encrypted checksum ck.

He checks that a, b, c and h are valid. This ensures they are linear combinations of values from the CRS.

He checks the divisibility condition:

$$e(a, b) = e(c, E(1)) \cdot e(E(T(z)), h)$$

This proves that $A(z)\, B(z) = C(z) + T(z)\, H(z)$.

He checks that

$$e(ck, E(\delta)) = e(a, E(\alpha\delta)) \cdot e(b, E(\beta\delta)) \cdot e(c, E(\gamma\delta))$$

This proves that A, B, C are linear combinations of the A_i, B_i, C_i with the same coefficients s_i.

(Non-obvious consequence of the KEA.)

Summary

Summary

Sigma protocols:

- The main “design pattern” to describe and analyze interactive ZK protocols.
- Transformation to non-interactive protocols via the Fiat-Shamir heuristic.

ZK-SNARK proofs:

- Able to prove the knowledge of a solution to a wide class of problems.
- The QAP approach: arithmetic circuits encoded as poynomials.
- Other approaches are possible, see the survey by Nitulescu (2020).

References

References

Sigma protocols:

- *Cryptography made simple*, Nigel P. Smart, Springer, 2016. Chapter 21.

A survey on ZK-SNARKS:

- Anca Nitulescu, *zk-SNARKs: A Gentle Introduction*, 2020. https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf

A step-by-step tutorial on the QAP approach:

- Maksym Petkus, *Why and How zk-SNARK Works: Definitive Explanation*, 2019, http://arxiv.org/abs/1906.07221

An overview of QAP and its applications to blockchains:

- Thomas Chen, Hui Lu, Teeramet Kunpittaya, Alan Luo, *A Review of zk-SNARKs*, 2023, https://arxiv.org/abs/2202.06877