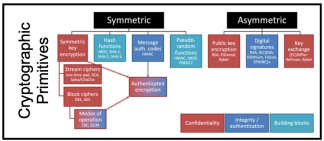
Topic 4.5
Applications – Zero knowledge

Douglas Stebila

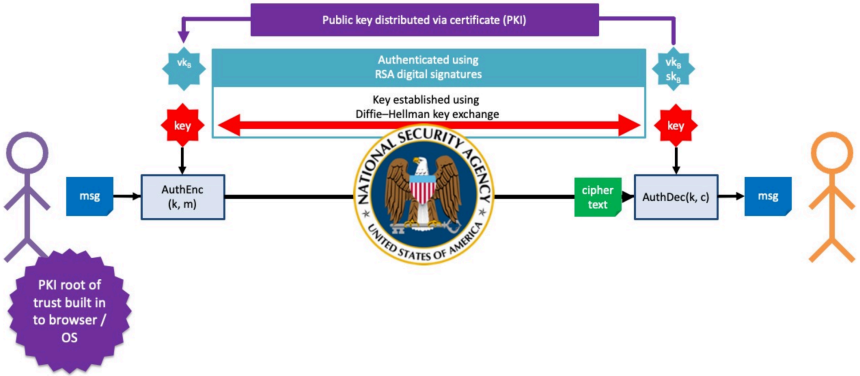
CO 487/687: Applied Cryptography
Fall 2024

Applications



Key management	Secure channels	Other applications
• PKI • X.509 certificates	• TLS • SSH • Signal	• Bitcoin • Zero knowledge

Secure channels



Zero knowledge proofs

Zero-knowledge proof: Peggy wants to prove to Victor that a certain statement is true, without disclosing anything other than the truth of the statement.

Zero-knowledge proof of knowledge: Peggy wants to prove to Victor that she knows something, without disclosing any information about the thing she knows.

Example: Two balls and the colour-blind friend

Imagine Peggy wants to prove to her colour-blind friend Victor that two balls are different colours, without telling him which one is which colour.

They play the following game:

1. Victor puts the two balls behind his back, then brings out ball out and displays it to Peggy
2. Victor puts both balls behind his back again. He flips a private coin; if it's heads, he bring the same ball out again; if it's tails, he brings out the other ball
3. Peggy tells him whether he brought out the same ball again or a different ball

https://en.wikipedia.org/wiki/Zero-knowledge_proof#Two_balls_and_the_colour-blind_friend

Example: Two balls and the colour-blind friend

Key idea: Victor can't tell if the balls are different colours or the same colour, but he does know whether he is bringing out the same ball again or the other ball.

- Zero-knowledge: Victor doesn't learn which ball is which colour.
- Completeness: If the balls are different colours, then Peggy can convince Victor: Peggy can always distinguish the balls with certainty and answer Victor's challenge correctly.
- Soundness: If the balls are the same colour, then Peggy can't convince Victor except with small probability. In any attempt, Peggy can only guess with 50% probability whether Victor switched the balls or not. If Victor repeats the game n times, the chance that Peggy fools Victor all n times is $1/2^n$.

https://en.wikipedia.org/wiki/Zero-knowledge_proof#Two_balls_and_the_colour-blind_friend

Desired properties

Completeness:

- If the statement is true, then an honest prover Peggy (who follows the protocol) can convince an honest verifier Victor that the statement is true.

Soundness:

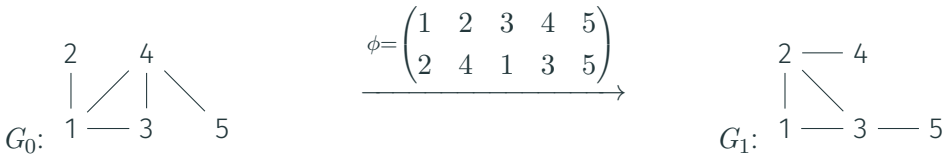
- If the statement is false, then no honest or cheating prover Peggy can convince an honest verifier Victor that it is true, except with small probability.
  - It is okay if the verifier Victor rejects with at least constant probability, not probability 1.
  - Sometimes hard to achieve high probability with one execution.
  - But can repeat the protocol many times to amplify soundness probability: with high probability, Victor will reject at least once in many repetitions.

Zero knowledge:

- If the statement is true, then no verifier learns anything other than the fact that the statement is true.

Graph isomorphism

Let G_0 and G_1 be graphs. An isomorphism $\phi : G_0 \to G_1$ is a relabelling (permutation) of the vertices of G_0 that yields the graph G_1.



Zero-knowledge proof of knowledge of graph isomorphism

Peggy wants to prove to Victor that she knows an isomorphism ϕ between two public graphs G_0, G_1, without disclosing any information about ϕ to Victor.

Proof of knowledge using commitment–challenge–response:

1. Peggy generates a commitment and sends it Victor
2. Victor generates a challenge and sends it to Peggy
3. Peggy generates a response and sends it to Victor
4. Victor verifies the response

Zero-knowledge proof of knowledge of graph isomorphism

Commitment:

- Peggy generates a random secret permutation ψ and computes $H = \psi(G_1)$.
- Commitment is H

Challenge:

- Victor picks a random bit $b \in_R \{0, 1\}$; Challenge is b

Response:

- Peggy's response will be the isomorphism between G_b and H:
  - If $b = 0$: Response is $\chi = \psi \circ \phi$
  - If $b = 1$: Response is $\chi = \psi$

Verification:

- Victor checks that $\chi(G_b) = H$

Completeness and soundness for graph isomorphism

Completeness:

- Three isomorphisms:
  - $\phi : G_0 \to G_1$
  - $\psi : G_1 \to H$
  - $\psi \circ \phi : G_0 \to H$
- So Peggy's response $\chi = \begin{cases} \psi \circ \phi, & \text{if } b = 0 \\ \psi, & \text{if } b = 1 \end{cases}$ will indeed map G_b to H

Soundness:

- If Peggy does not know $\phi : G_0 \to G_1$, she could guess what b Victor will use then generate a random χ and compute $H = \chi(G_b)$.
- But she will only guess b right 50% of the time.
- The other half of the time she'll be wrong and verification will fail.
- Soundness probability: $\frac{1}{2}$.

Desired properties

Zero knowledge:

- If the statement is true, then no verifier learns anything other than the fact that the statement is true.
- How to formalize “learns nothing”?
- Victor learns nothing if he could have generated all of the values he receives on his own.
- In other words, if there exists a simulator that outputs transcripts that are indistinguishable from real transcripts
- But Victor may have to generate them in a different order:
  - Honest execution: 1) generate commitment, 2) receive challenge, 3) compute response
  - Simulation: 1) pick challenge, 2) generate response, 3) retroactively compute commitment

Zero knowledge for graph isomorphism

Simulator to demonstrate zero knowledge:

1. Pick random $b \in_R \{0, 1\}$
2. Pick random isomorphism χ
3. Compute $H = \chi(G_b)$

This transcript (H, b, χ) has the same distribution as real transcripts. So when Victor sees real transcripts, he isn't learning any information he couldn't have computed himself.

But does this mean Peggy can forge transcripts too?

- No! Order matters.
  - Peggy has to make a commitment H that will work for any challenge.
  - The simulator can retroactively build the commitment to work for one particular challenge.

Schnorr's identification scheme

Scenario: Peggy wants to prove to Victor that she knows the secret key behind a public key.

Let G be a group of prime order q with generator g.

Peggy's secret key: $x \in_R \mathbb{Z}_q$
Peggy's public key: $y = g^x$

Goal: Proof of knowledge of x s.t. $y = g^x$.

Commitment:

- $k \in_R \mathbb{Z}_q$; $r \leftarrow g^k$
- Commitment is r

Challenge:

- $e \in_R \mathbb{Z}_q$; Challenge is e

Response:

- $s \leftarrow k + xe \bmod q$, Response is s

Verification:

- Check that $g^s y^{-e} = r$

Properties of Schnorr identification

Completeness:

- Substitute and check equation

Soundness:

- If Peggy does not know the discrete logarithm x, her best chance of success is guessing a random response, which is successful with probability $1/q$
- Also has a technical property called “special soundness”

Zero-knowledge:

- Simulator:
  1. Pick $e \in_R \mathbb{Z}_q$
  2. Pick $s \in_R \mathbb{Z}_q$
  3. Compute $r = g^s y^{-e}$
- Has the same distribution has honest transcripts, but **generated in a different order**.

Non-interactive proofs

In most proof of knowledge protocols, the challenge is picked randomly.
Can we make the protocol non-interactive and have the prover compute everything?

If prover can pick commitment after challenge, then possible to fool the verifier.
(This is how our zero knowledge simulations worked!)

Idea: challenge = hash of commitment

Called the Fiat–Shamir transform.

- Technically: challenge = hash of commitment and statement
- Secure assuming the hash is a random function

Schnorr signature scheme

Idea: Use the Fiat–Shamir transform to turn the interactive Schnorr identification protocol into a non-interactive signature scheme.

Key generation:

- Secret key: $x \in_R \mathbb{Z}_q$
- Public key: $y = g^x$

Sign($sk = x, m \in \{0,1\}^*$):

1. Pick $k \in_R \mathbb{Z}_q$
2. $r \leftarrow g^k$
3. $e \leftarrow H(r \| m)$
4. $s \leftarrow k + xe \bmod q$
5. Return signature (r, s)

Verify($pk = y, m, \sigma = (r, s)$):

1. $e \leftarrow H(r \| m)$
2. Check that $g^s y^{-e} = r$

Theorem: If the identification scheme is secure against impersonation attacks against a passive adversary, and H is a random function, then the signature scheme constructed via the Fiat–Shamir transform is existentially unforgeable under chosen message attacks.

Proof of discrete logarithm equality

Scenario: Peggy wants to prove to Victor that $\mathrm{DLOG}_g(y_1) = \mathrm{DLOG}_h(y_2)$.

(In other words, that there exists x such that $g^x = y_1$ and $h^x = y_2$ simultaneously.)

Chaum–Pedersen Protocol for proof of discrete logarithm equality

Goal: Proof that $\mathrm{DLOG}_g(y_1) = \mathrm{DLOG}_h(y_2)$.

Commitment:

- $k \in_R \mathbb{Z}_q$
- $r_1 \leftarrow g^k$, $r_2 \leftarrow h^k$
- Commitment is (r_1, r_2)

Challenge:

- $e \in_R \mathbb{Z}_q$; Challenge is e

Response:

- $s \leftarrow k - xe \bmod q$, Response is s

Verification:

- Check that $g^s y_1^e = r_1$ and $h^s y_2^e = r_2$

Dilithium / ML-DSA signature scheme

ML-DSA is NIST's newly standardized post-quantum signature scheme.

Main idea:

- Zero-knowledge proof of knowledge of a module learning with errors (LWE) secret key (s, e) for public key (A, b) where $b = As + e$
- Use Fiat–Shamir transform to convert interactive proof into non-interactive signature scheme

More proofs of knowledge

- Prove that some linear relation of discrete logarithms is satisfied (not just equality)
- Prove that a number is in a range
- Prove that a number is composite
- Proving knowledge of one of several things
- Can construct a zero-knowledge proof system for any language in NP
- Zero-knowledge against an honest verifier versus malicious verifier
- zk-SNARKs: zero knowledge succinct non-interactive arguments of knowledge: used in Ethereum, Zcash, other cryptocurrencies
- STARKs: T=“transparent”: SNARKS without trusted setup