Radbound University

Commitment schemes, Identification Protocols and Fiat-Shamir Signatures

Applied Cryptography – Spring 2024

Simona Samardjiska

April 15, 2024

Institute for Computing and Information Sciences
Radboud University

Summary

Last time:

- Public key encryption
- Key Encapsulation Mechanisms
- Digital Signatures from trapdoor permutations

Today:

- Commitment schemes
- Zero-Knowledge protocols
- Identification Protocols
- Fiat-Shamir Signatures
- DSA and ECDSA

Commitment schemes

Coin flipping by telephone (Blum)

- Alice and Bob are getting a divorce!
- They are at the point where they cannot even stand facing each other, so they have to discuss over the phone how to split the furniture, the kids, etc.
- But one problem remains: who gets the car?
- They decide to flip a coin over the phone. . .
- **First attempt**:

Alice		Bob
	$\xrightarrow{\text{You guess, I flip}}$	
	$\xleftarrow{b_B}$	$b_B \overset{\$}{\leftarrow} \{heads, tails\}$
$b_A \overset{\$}{\leftarrow} \{heads, tails\}$		
If $b_A \overset{?}{=} b_B$ Bob wins		
otherwise, Alice wins		

- **Wait a minute, Alice can cheat!**

Coin flipping by telephone (Blum)

- **Second attempt**:

$$
\begin{array}{lcl}
\text{Alice} & & \text{Bob} \\
\hline
 & \xleftarrow{\text{You flip, I guess}} & \\
b_A \xleftarrow{\$} \{\mathit{heads}, \mathit{tails}\} & \xrightarrow{b_A} & \\
 & \xleftarrow{b_B} & b_B \xleftarrow{\$} \{\mathit{heads}, \mathit{tails}\} \\
\text{If } b_A \stackrel{?}{=} b_B \text{ Bob wins} & & \\
\text{otherwise, Alice wins} & &
\end{array}
$$

- **Wait a minute, Bob can cheat!**
- A deadlock! Any ideas?

Coin flipping by telephone (Blum)

- **Third attempt: Use a commitment** (a locked box):
  - Commit to value $c = commit(b)$
  - Reveal (open) value $b = open(c)$

Alice		Bob
$b_A \xleftarrow{\$} \{heads, tails\}$		
$c_A = commit(b_A)$	$\xrightarrow{c_A\text{(commit)}}$	
	$\xleftarrow{b_B}$	$b_B \xleftarrow{\$} \{heads, tails\}$
	$\xrightarrow{b_A\text{(open)}}$	
If $b_A \stackrel{?}{=} b_B$ Bob wins otherwise, Alice wins		If $b_A \stackrel{?}{=} b_B$ Bob wins otherwise, Alice wins

- When does this protocol work?
  - If Alice can find another b'_A such that $commit(b'_A) = commit(b_A)$, then **Alice can cheat!** (*commit* **should be binding!**)
  - If Bob can find b_A given $commit(b_A)$, then **Bob can cheat!** (*commit* **should be hiding!**)

Commitment scheme – formally

Given security parameter $\lambda \in \mathbb{N}$ and two finite sets $\mathcal{M}, \mathcal{R} \subseteq \{0,1\}^*$, a commitment scheme $\mathsf{Comm} = (\mathsf{Setup}, \mathsf{Comm}, \mathsf{Open})$ consists of three algorithms:

- **Setup** $\mathsf{pk} \leftarrow \mathsf{Setup}(1^\lambda)$
- **Commitment algorithm**: Takes random $r \in \mathcal{R}$, the message $M \in \mathcal{M}$ and outputs $(C, D) \leftarrow \mathsf{Comm}(\mathsf{pk}, M, r)$. C is said to be the commitment of M, and D is the opening (decommitment).
- **Opening (verification) algorithm**: Takes as input the commitment of M, the opening D, and outputs $M' = \mathsf{Open}(\mathsf{pk}, C, D) \in \mathcal{M}$ or $\perp \notin \mathcal{M}$ to indicate an invalid commitment.

Correctness: For all $M \in \mathcal{M}$

$$\mathsf{Open}(\mathsf{pk}, (\mathsf{Comm}(\mathsf{pk}, M, r))) = M$$

Note:

- Often the opening simply contains the message M and the random coins r used in the commitment generation,
- Now, the verification consists of running again the Comm algorithm and checking whether it matches

Commitment scheme – properties

Binding property: The sender can not change their mind after committing

- **Unconditional/statistical binding:** Even with an infinite computational power, it is not possible to change mind! In this case b is uniquely determined/is determined with overwhelming probability by $\mathrm{Comm}(\mathrm{pk}, b, r)$
- **Computational binding:** (Limited to computationally bounded senders.) For every PPT algorithm $\mathcal{A}$ the probability of finding two different valid openings of a commitment is negligible.

Hiding property: a commitment to b reveals (almost) no information about b

- **Unconditional/statistical hiding:** for every $M_0, M_1 \in \mathcal{M}$ the two distribution ensembles

$$\{C_0 | (C_0, D_0) \leftarrow \mathrm{Comm}(\mathrm{pk}, M_0, R_0)\} \quad \text{and} \quad \{C_1 | (C_1, D_1) \leftarrow \mathrm{Comm}(\mathrm{pk}, M_1, R_1)\}$$

  are identical/statistically indistinguishable, i.e. the statistical distance between the two is zero/negligible.
- **Computational hiding:** (Limited to computationally bounded receivers.) For every PPT algorithm $\mathcal{A}$ and every $M_0, M_1 \in \mathcal{M}$ the two distribution ensembles above are computationally indistinguishable.

Commitment scheme – examples

Perfectly binding and perfectly hiding commitment:

- perfectly binding $\Rightarrow$ there exist no different R_0, R_1, M_0, M_1 s.t. $\mathrm{Comm}(\mathrm{pk}, M_0, R_0) = \mathrm{Comm}(\mathrm{pk}, M_1, R_1)$
- $\Rightarrow$ An unbounded adversary can always find M given $\mathrm{Comm}(\mathrm{pk}, M, R)$ (by brute-forcing all M and R)
- $\Rightarrow$ Such commitment scheme **does not exist!**

Hash based commitment:

$$\mathrm{Comm}(\mathrm{pk}, M, R) = H(R||M) \text{ where } R \xleftarrow{\$} \{0,1\}^{t\lambda}$$

- Computationally binding if H- collision resistant
- Computationally hiding if H- preimage resistant

  In the ROM:
- If message length is bounded - statistical binding can be achieved
- Statistical hiding can be achieved for $t \geqslant 5$

Commitment scheme – examples

Pedersen bit-commitment:

$$\mathrm{Comm}(\mathrm{pk}, B, R) = g^R \cdot h^B \text{ where } R \xleftarrow{\$} \mathbb{Z}_n \text{ and } h \in \langle g \rangle, |\langle g \rangle| = n$$

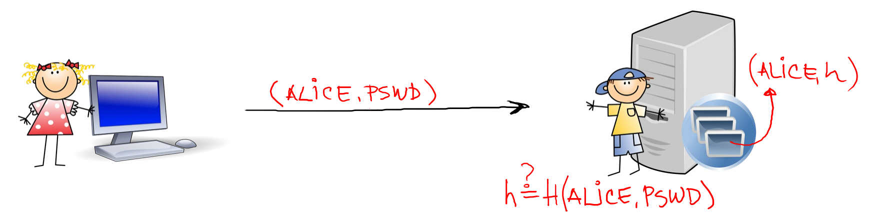
h is such that $\log_g h$ is unknown to the parties

- Computationally binding under DL assumption
  - $\mathrm{Comm}(\mathrm{pk}, B, R_0) = \mathrm{Comm}(\mathrm{pk}, 1 - B, R_1) \Rightarrow g^{R_0} \cdot h^B = g^{R_1} \cdot h^{1-B}$
  - $\Rightarrow \log_g h = (R_0 - R_1)/(1 - 2B)$
  - $\Rightarrow$ DL broken!
- Perfectly hiding - g^R and $g^R h$ are perfectly indistinguishable

Zero Knowledge Protocols

In practice ...

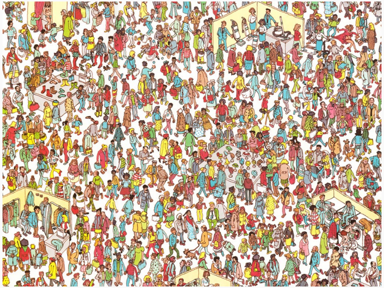
- Most common type of authentication - using passwords
- Typical flow:



- Many things can go wrong when password is sent to server and hash stored at server
  - insecure communication
  - insecure storage
- Alice would rather not send her password... but still be able to prove she knows it
- Solution?

Zero Knowledge Identification Protocols

- Alice (the prover $\mathcal{P}$) wants to prove the knowledge of a secret to Bob (the verifier $\mathcal{V}$)
- They engage in an **interactive proof** protocol
- Important:
  - Alice should (almost) always be able to convince Bob if she knows the secret (completeness)
  - Alice should (almost) never be able to convince Bob if she doesn't know the secret, even if she cheats (soundness)
  - The interaction should leak the secret as little as possible even if Bob cheats
    - Ideally, not at all - **Zero-Knowledge (ZK) property**



Interactive proof/argument systems

Let L be a language. The pair $(\mathcal{P}, \mathcal{V})$ is an interactive proof system for L (proving membership of statement) if it satisfies the following two conditions:

- Completeness: If $x \in L$, then the probability that $(\mathcal{P}, \mathcal{V})$ rejects x is negligible in the length of x.
- Soundness: If $x \notin L$ then for any prover $\mathcal{P}^*$, the probability that $(\mathcal{P}^*, \mathcal{V})$ accepts x is negligible in the length of x. This probability is called **soundness error**.

Some terminology:

- **Cheating parties** will be denoted by $\mathcal{P}^*$, $\mathcal{V}^*$ - they don't follow the protocol (may use a cheating strategy)
- **Transcript** - a record of the entire conversation between the parties
- The verifier is always polynomial time bounded
- The prover may be unbounded (proof systems) or polynomial time bounded (argument systems)
- In practice we will be always interested in **arguments**

What does Zero Knowledge mean?

- Whatever strategy the verifier follows, and whatever a priori knowledge he may have, he learns nothing except for the truth of the prover's claim
- if nothing is leaked, this means that the verifier can **simulate** the conversation with the prover without even interacting with him!

An interactive proof system or argument $(\mathcal{P}, \mathcal{V})$ for language L is **zero-knowledge** if for every PPT verifier $\mathcal{V}^*$, there is a simulator $\mathcal{S}_{\mathcal{V}^*}$ running in expected probabilistic polynomial time, such that:
For $x \in L$ the distribution of transcripts output by $\mathcal{S}_{\mathcal{V}^*}$ on input x is

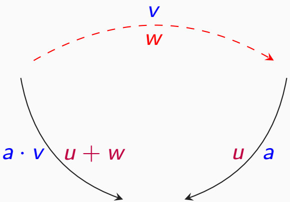
- perfectly (perfect ZK)
- statistically (statistical ZK)
- computationally (computational ZK)

indistinguishable from the distribution of transcripts produced by $(\mathcal{P}, \mathcal{V}^*)$ on input x (given to $\mathcal{V}^*$).

- the simulator can generate messages of a transcript **in any order it wants**

An example walkthrough - Schnorr protocol

- Given p-prime, $q|p-1$, $g \in \mathbb{Z}_p^*$ of order q, $w \in \mathbb{Z}_q$ and $v = g^w \pmod{p}$.
- The prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of w without revealing any information about it



- Inside of circle - addition
- Outside of circle - multiplication
- Prover shows they can close circle, by revealing **only one** of purple values

$\mathcal{P}(\mathsf{pk} = v, \mathsf{sk} = w)$		$\mathcal{V}(\mathsf{pk} = v)$
$u \xleftarrow{\$} \mathbb{Z}_q, a \leftarrow g^u$		
$\mathsf{com} \leftarrow a$	$\xrightarrow{\mathsf{com}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
$\mathsf{resp} \leftarrow u + \mathsf{ch} \cdot w$	$\xrightarrow{\mathsf{resp}}$	
		$g^{\mathsf{resp}} \stackrel{?}{=} a \cdot v^{\mathsf{ch}}$

An example walkthrough - Schnorr protocol

- **Completeness:** Trivially satisfied. If indeed $v = g^w$ and $a = g^u$, then the verifier will always accept. **Why?**
- **Soundness:** Suppose the prover does not know w. $\mathcal{P}$ can always answer one challenge, if they prepare well. **How?**
  - $\mathcal{P}$ makes a guess which challenge they will receive, say ch^*, and they prepare for it by calculating $\mathrm{resp} \xleftarrow{\$} \mathbb{Z}_q$ and then $a = g^{\mathrm{resp}}/v^{\mathrm{ch}^*}$.
  - $\mathcal{P}$ commits to a.
  - If the real challenge $\mathrm{ch} = \mathrm{ch}^*$, the verifier accepts, otherwise it rejects.

  $\Rightarrow \mathcal{P}$ convinces the verifier with probability $1/2$ - soundness error.

  Can $\mathcal{P}$ succesfully answer both challenges?
  - Suppose they can. Let resp_0 and resp_1 be the two correct answers to the challenges 0 and 1.
  - Then we get $w = \mathrm{resp}_1 - resp_0$, which is a contradiction – the prover can easily calculate w.

To reduce the error to negligible in a security parameter λ, the protocol needs to be repeated λ times. The soundness error becomes $1/2^\lambda$.

An example walkthrough - Schnorr protocol

- **Zero-Knowledgeness:** We will show the protocol is zero-knowledge. We build a simulator $\mathcal{S}$ (no knowledge of the secret w, and is polynomially bounded) as follows:
  1. $\mathcal{S}$ starts the verifier $\mathcal{V}^*$ by giving it the parameters p, q, g and the public v
  2. $\mathcal{S}$ makes a guess which challenge it will receive, say ch^*, and it prepares for it by calculating $a = g^{\mathsf{resp}}/v^{\mathsf{ch}^*}$ for randomly chosen $\mathsf{resp} \xleftarrow{\$} \mathbb{Z}_q$. $\mathcal{S}$ sends $\mathsf{com} = a$ to $\mathcal{V}^*$.
  3. $\mathcal{S}$ gets a challenge ch from $\mathcal{V}^*$. If $\mathsf{ch} = \mathsf{ch}^*$, $\mathcal{S}$ outputs $(a, \mathsf{ch}^*, \mathsf{resp})$. If $\mathsf{ch} \neq \mathsf{ch}^*$, $\mathcal{S}$ rewinds the verifier $\mathcal{V}^*$ to the point before it receives com, and goes to step 2.

- **Rewinding of adversaries/cheating parties**: We always consider these to be probabilistic Turing machines, so rewinding is possible, until a desired output is produced
- **$\mathcal{S}$ is expected probabilistic polynomial time:** One round is expected to be repeated 2 times, hence whole simulation takes expected 2λ time.
- **Distributions of simulated and real protocol are exactly the same**:The candidate commitments a are uniform over the group $\langle g \rangle$, as well as the responses, just as in the real protocol. The challenge is produced just as in the real protocol (it is produced by $\mathcal{V}^*$). Hence the distributions are identical (we say the protocol is <u>perfect</u> zero-knowledge.)

Homework - ZK for Graph Isomorphism (GI)

- Let ϕ be an isomorphism between two graphs $\mathcal{G}_0$ and $\mathcal{G}_1$ s.t. $\mathcal{G}_1 = \phi(\mathcal{G}_0)$.
- Given $\mathcal{G}_0, \mathcal{G}_1$, the prover $\mathcal{P}$ wants to prove to the verifier $\mathcal{V}$ knowledge of ϕ without revealing any information about it

$$\begin{tikzcd} \mathcal{G}_0 \arrow[r, "\phi_0"] \arrow[d, dashed, "\phi"'] & \mathcal{G}' \\ \mathcal{G}_1 \arrow[ur, "\phi_1"'] & \end{tikzcd}$$

$\mathcal{P}(\mathcal{G}_0, \mathcal{G}_1, \phi)$		$\mathcal{V}(\mathcal{G}_0, \mathcal{G}_1)$
com $\leftarrow \mathcal{G}'$	$\xrightarrow{\text{com}}$	
	$\xleftarrow{\text{ch}}$	ch $\xleftarrow{\$} \{0, 1\}$
resp $\leftarrow \phi_{\text{ch}}$	$\xrightarrow{\text{resp}}$	
		$\mathcal{G}' \stackrel{?}{=} \phi_{\text{ch}}(\mathcal{G}_{\text{ch}})$

Note:

- No probabilistic polynomial-time algorithms are known for the GI problem
- This generalizes to other isomorphisms on different objects
- **Homework: Show in detail that the above protocol is Zero-Knowledge!**

Applications of ZK proofs

- Whenever we need to prove knowledge of secrets without revealing them
- Hence, for protecting confidentiality, privacy, anonymity
- Indispensible tool in Privacy-enhancing technologies (PETs)

Concrete applications:

- For anonymous, verifiable voting
  - voters can be sure their vote is anonymous (their identity is not connected to the cast vote) and that their vote is included in the tally.
- For user authentication
  - as identification schemes without revealing or exchanging the passwords
- In multi-party computation
  - to make sure parties don't cheat and follow the protocol specs
- For preserving privacy of data
  - for example, to show to the bank you have enough income to repay a loan, without revealing the actual income
- In Blockchain technologies
  - for privacy and anonymity

Sigma protocols

Σ protocols

$$
\begin{array}{lcl}
\mathcal{P}, \mathsf{sk} & & \mathcal{V}, \mathsf{pk} \\
\hline
\mathsf{com} \xleftarrow{\$} \mathcal{P}_0(\mathsf{sk}) & & \\
 & \xrightarrow{\quad \mathsf{com} \quad} & \\
 & & \mathsf{ch} \xleftarrow{\$} \mathsf{ChS}(1^k) \\
 & \xleftarrow{\quad \mathsf{ch} \quad} & \\
\mathsf{resp} \leftarrow \mathcal{P}_1(\mathsf{sk}, \mathsf{com}, \mathsf{ch}) & & \\
 & \xrightarrow{\quad \mathsf{resp} \quad} & \\
 & & b \leftarrow \mathsf{Vf}(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})
\end{array}
$$

Given a relation $R = V \times W$, where sk $\in W$ is called **witness** and language $L_R = \{v \in V | \exists w \in W, (v, w) \in R\}$ for the relation, a Σ protocol is a three move protocol as above satisfying:

- **Completeness**: (as before)
- **Special soundness**: There exists a PPT algorithm $\mathcal{K}$ - knowledge extractor, that given two valid transcripts $\mathsf{trans} = (\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, $\mathsf{trans}' = (\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$, $\mathsf{ch} \neq ch'$, extracts the witness sk with non-negligible probability
- **Special Honest Verifier ZK**: Same as before, except, the verifier is honest (follows the protocol), and the simulator $\mathcal{S}$ needs to output a valid transcript for a given challenge ch.

Recall Schnorr's protocol

p-prime, $q|p-1$, $g \in \mathbb{Z}_p^*$ of order q. $w \in \mathbb{Z}_q$ and $v = g^w \pmod{p}$

$\mathcal{P}, \mathsf{sk} = w$		$\mathcal{V}, \mathsf{pk} = v$
$u \xleftarrow{\$} \mathbb{Z}_q, a \leftarrow g^u$	$\xrightarrow{\quad a \quad}$	
	$\xleftarrow{\quad \mathsf{ch} \quad}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
$\mathsf{resp} \leftarrow u + \mathsf{ch} \cdot w$	$\xrightarrow{\quad \mathsf{resp} \quad}$	
		$g^{\mathsf{resp}} \stackrel{?}{=} a \cdot v^{\mathsf{ch}}$

- **Soundness error** - $1/2$, so needs many rounds to achieve negligible soundness error
- Each round performs expensive exponentiations in a group of order q
- Can we do better?

A more efficient **single round variant** - **Schnorr identification protocol**

$\mathcal{P}, \mathsf{sk} = w$		$\mathcal{V}, \mathsf{pk} = v$
$u \xleftarrow{\$} \mathbb{Z}_q, a \leftarrow g^u$		
	$\xrightarrow{\quad a \quad}$	
		$\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_q$
	$\xleftarrow{\quad \mathsf{ch} \quad}$	
$\mathsf{resp} \leftarrow u + \mathsf{ch} \cdot w$		
	$\xrightarrow{\quad \mathsf{resp} \quad}$	
		$g^{\mathsf{resp}} \stackrel{?}{=} a \cdot v^{\mathsf{ch}}$

- We show Schnorr identification protocol is Σ-protocol
- **Completeness**: Prover that knows w always succeeds

An example walkthrough - Schnorr protocol

- **Special soundness**: Given two accepting transcripts for the same commitment $\mathsf{trans} = (\mathsf{com}, \mathsf{ch}, \mathsf{resp}) = (a, \mathsf{ch}, u + \mathsf{ch} \cdot w)$, and $\mathsf{trans}' = (\mathsf{com}, \mathsf{ch}', \mathsf{resp}') = (a, \mathsf{ch}', u + \mathsf{ch}' \cdot w)$, we have

$$w = \frac{\mathsf{resp} - \mathsf{resp}'}{(\mathsf{ch} - \mathsf{ch}')} \qquad \Rightarrow \text{ the witness can be extracted with probability 1.}$$

- **Not known to be ZK**: (but no known attacks)
  - One round implies that the challenge space must be exponentially large
  - This implies "rewinding until challenge is guessed" requires exponential time
  - This implies PPT simulator as previously will fail!
- **Special HVZK**: For given $\mathsf{ch} \in \mathbb{Z}_q$
  - Choose $\mathsf{resp} \xleftarrow{\$} \mathbb{Z}_q$ and calculate $a \leftarrow g^{\mathsf{resp}} v^{-\mathsf{ch}}$
  - The distributions of the real transcripts and the simulated transcripts are the same – in both a given valid transcript occurs with prob. $1/q$ (in one first u is chosen a random, in the other first resp is chosen at random.)

Fiat-Shamir signatures

The Fiat-Shamir transform

Σ-protocol

$\mathcal{P}(\mathsf{pk}, \mathsf{sk})$		$\mathcal{V}(\mathsf{pk})$
$\mathsf{com} \leftarrow \mathcal{P}_0\ (\mathsf{sk})$	$\xrightarrow{\mathsf{com}}$	
$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}\ (1^k)$	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow \mathcal{P}_1\ (\mathsf{sk}, \mathsf{com}, \mathsf{ch})$	$\xrightarrow{\mathsf{resp}}$	
		$b \leftarrow \mathsf{Vf}\ (\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

$\downarrow \quad \downarrow \quad \downarrow$

FS signature

Signer
$\mathsf{com} \leftarrow \mathcal{P}_0{}^r(\mathsf{sk})$
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$\mathsf{resp} \leftarrow \mathcal{P}_1{}^r(\mathsf{sk}, \mathsf{com}, \mathsf{ch})$
output : $\sigma = (\mathsf{com}, \mathsf{resp})$

Verifier
$\mathsf{ch} \leftarrow H(m, \mathsf{com})$
$b \leftarrow \mathsf{Vf}^r(\mathsf{pk}, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$
output : b

The Fiat-Shamir transform

Let $\lambda \in \mathbb{N}$ the security parameter, $\mathsf{IDS} = (\mathsf{KGen}, \mathcal{P}, \mathcal{V})$ an identification scheme that is a Σ-protocol (special sound with knowledge error κ and HVZK).

Let $H : \{0,1\}^* \to \{0,1\}^r$ be modelled as a random oracle. The **Fiat-Shamir signature scheme** derived from IDS is the triplet of algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ s.t.:

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^k)$,
- $\sigma = (\mathsf{com}, \mathsf{resp}) \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ where $\mathsf{com} \leftarrow \mathcal{P}_0^r(\mathsf{sk})$, $h = H(m, \mathsf{com})$, $\mathsf{resp} \leftarrow \mathcal{P}_1^r(\mathsf{sk}, \mathsf{com}, h)$.
- $\mathsf{Vf}(\mathsf{pk}, m, \sigma)$ parses $\sigma = (\mathsf{com}, \mathsf{resp})$, computes $h = H(m, \mathsf{com})$, and outputs $\mathcal{V}^r(\mathsf{pk}, \mathsf{com}, h, \mathsf{resp})$.

- The number of rounds r is chosen such that $\kappa^r < \frac{1}{2^\lambda}$.
- Note that the "full" signature is $(\mathsf{com}, h, \mathsf{resp})$, but h can be omitted, since it can be recreated from m and com.

Security of FS signatures [Pointcheval & Stern '96]

Structure of proof:

IDS		**FS signature**
Special soundness	$\longrightarrow$	Secure under key-only attack
+	$\searrow$	
HVZK	$\longrightarrow$	**eu-cma secure**

Proof in ROM (see additional literature if interested!)

Schnorr signature

KeyGen:

1. Choose two primes p, q s.t. $q|p-1$, and $g \in \mathbb{Z}_p^*$ of order q.
2. Choose a random $w \in \mathbb{Z}_q$ and compute $v = g^w \pmod p$
3. Output public key $\mathsf{pk} = v$ and private key $\mathsf{sk} = w$

Sign: Given message M,

1. $u \xleftarrow{\$} \mathbb{Z}_q, \mathsf{com} \leftarrow g^u$,
2. Set $\mathsf{ch} = H(M, \mathsf{com})$ and calculate $\mathsf{resp} \leftarrow u + \mathsf{ch} \cdot w$
3. Set $\sigma = (\mathsf{com}, \mathsf{resp})$
4. Output message - signature pair (M, σ)

Verify: To verify the message - signature pair (M, σ)

1. Parse $\sigma = (\mathsf{com}, \mathsf{resp})$ and calculate $\mathsf{ch} = H(M, \mathsf{com})$
2. Check $g^{\mathsf{resp}} \stackrel{?}{=} a \cdot v^{\mathsf{ch}}$ and output Accept if check succeeds, otherwise Fail

Schnorr signature and Digital Signature Algorithm (DSA)

- Schnorr signature proposed in 1990
  - simple, efficient, provably secure,unfortunately, patented (1990-2010)
- In 1990's RSA signature also patented
- NIST proposes in 1991 the Digital Signature Algorithm (DSA)
  - NIST standard from 1994 - Federal standard (FIPS 186) and ANSI X9.30 Part 1
  - Very similar to Schnorr, but initially no proof!!!
  - Modified versions under suitable models later shown to be provably secure ...
  - In ISO/IEC 14888, $h = H(M)$ replaced by $h = H(M, r)$
  - ECDSA - elliptic curve version over the elliptic curve group $E(\mathbb{Z}_p)$
    - additive notation, otherwise same as DSA (see assignment)
  - ECDSA is widely used today in blockchain, iOS, secure messaging apps, in TLS (but not even close to RSA signatures)!
  - Faster signing, slower verification than RSA, significantly smaller keys

Digital Signature Algorithm (DSA)

KeyGen: Same as for Schnorr with private key $x \in \mathbb{Z}_q$ and public key $y = g^x \pmod{p}$

1. Choose two primes p, q s.t. $q | p - 1$, and $g \in \mathbb{Z}_p^*$ of order q.
2. Choose a random $x \in \mathbb{Z}_q$ and compute $y = g^w \pmod{p}$
3. Output public key $\mathsf{pk} = y$ and private key $\mathsf{sk} = x$

Sign: Given message M,

1. $k \xleftarrow{\$} \mathbb{Z}_q, r \leftarrow (g^k \pmod{p}) \pmod{q}$,
2. Set $h = H(M)$ and calculate $s \leftarrow (h + xr)k^{-1}$ (In Schnorr: $h = H(M, r)$ and $s \leftarrow k + h \cdot x$)
3. Set $\sigma = (r, s)$ and output message - signature pair (M, σ)

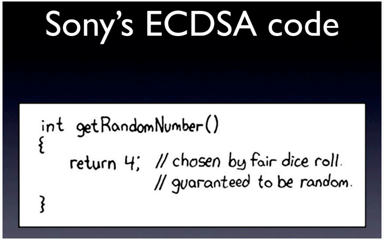
Verify: To verify the message - signature pair (M, σ)

1. Parse $\sigma = (r, s)$, verify $0 < r, s < q$ and calculate $h = H(M)$
2. Compute $u_1 = H(m)s^{-1} \pmod{q}$ and $u_2 = rs^{-1} \pmod{q}$
3. Check $(g^{u_1} g^{u_2} \pmod{p}) \pmod{q} \stackrel{?}{=} r$ and output Accept if check succeeds, otherwise Fail

Sensitive security of DSA and ECDSA

- DSA and ECDSA are very sensitive regarding the ephimeral key k
- Must not be reused, and should be unpredictable for attackers
- In December 2010, a group calling itself fail0verflow announced recovery of ECDSA private key used by Sony to sign software for PlayStation 3
  - Sony's "epic fail": k was static instead of random!

- Suppose $s_1 = (H(M_1) + xr)k^{-1}$ and $s_2 \leftarrow (H(M_1) + xr)k^{-1}$ use same k for two messages M_1 and M_2
- Now: $s_1 - s_2 = (H(M_1) - H(M_2))k^{-1}$
- I.e., $k = (H(M_1) - H(M_2))(s_1 - s_2)^{-1}$
- Once k is known, the secret key can be easily derived: $x = (s_1 k - H(M_1))(r)^{-1}$



Recommendations for key sizes and usage by NIST, ECRYPT

Algorithm	Param.	Key length	Classical security	Usage
RSA-1024	N	1024	80 bits	disallowed for key transport
RSA-1024/DSA-1024	N/q	1024	80 bits	disallowed signature key gen./legacy signature verification
RSA-2048/DSA-2048	N/q	2048	112 bits	acceptable
RSA-3072/DSA-3072	N/q	3072	128 bits	recommended
RSA-7680/DSA-7680	N/q	7680	192 bits	long term
RSA-15360/DLP-15360	N/q	15360	256 bits	long term
ECDSA-160	p	80	80 bits	disallowed signature key gen./legacy signature verification
ECDSA-224	p	224	112 bits	acceptable
ECDSA-256	p	256	128 bits	recommended
ECDSA-384	p	384	192 bits	long term
ECDSA-512	p	512	256 bits	long term

Summary

Today:

- Commitments
- Zero-Knowledge protocols
- Identification Protocols
- Fiat-Shamir Signatures
- DSA and ECDSA

Next time:

- Post-Quantum Cryptography
- Hash-Based signatures