02239 Data Security
Privacy

Sebastian Mödersheim

October 9, 2024

Outline

Outline

1. **Why Privacy?**
2. Defining Privacy
3. Mixes
4. Zero-Knowledge Protocols

- “An honest person has nothing to hide!”

- “1984”

An Example [from Mark Ryan]

- Parents and their teenage daughter:
  - ★ The daughter wants to go out, but not tell the parents where she goes.
  - ★ The parents want to know where she is in case of any emergency.
  - ★ Both are legitimate interests!
  - ★ Danger: the parents may overreach – out of concern for their daughter – not respecting her privacy.

An Example [from Mark Ryan]

- Parents and their teenage daughter:
  - ★ The daughter wants to go out, but not tell the parents where she goes.
  - ★ The parents want to know where she is in case of any emergency.
  - ★ Both are legitimate interests!
  - ★ Danger: the parents may overreach – out of concern for their daughter – not respecting her privacy.
- Non-electronic solution:
  - ★ The daughter writes where she is going in a sealed letter.
  - ★ The parents can open the letter, but are compelled not to — unless there is an emergency
- Technical solution: with trusted hardware (e.g. a TPM)

Why Privacy?

Why not vote in public?

Why Privacy?

- Being observed, or believing to be observed, can have an influence on ones behavior.
  - ★ Feeling compelled/coerced to act according to others' expectations.
  - ★ Can mean a subtle restriction of the freedom.
- This is getting more sensitive than 20 years ago. Life leaves more traces in different IT systems. Technology allows for:
  - ★ cheap storage of data
  - ★ cheap evaluations of data
  - ★ cheap surveillance
- Protesting against a totalitarian regime, exchange information with other opposition members.

Outline

Defining Privacy

What *is* privacy really?

Several informal and semi-formal notions:

- Anonymity: the identity of the actors are protected
- Unlinkability: different actions of the same actor cannot be associated
- ...

Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!

Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!
- Inspiration from Cryptography: security formulated as equivalence notions:

$$crypt(k, 0) \sim crypt(k, 1)?$$

The intruder knows that the plain-text is either 0 or 1, the challenge is for him to tell correctly whether it is 0 or 1.

Formally Defining Privacy

- Difficult: it is not a classical secrecy problem!
  - ★ Example: poll where you can vote *yes* or *no*
  - ★ *yes* and *no* are values that the intruder knows, also the names of the voters may be no secrets.
  - ★ What is actually secret is: who voted *yes* and who voted *no*!
- Inspiration from Cryptography: security formulated as equivalence notions:

  $$crypt(k, 0) \sim crypt(k, 1)?$$

  The intruder knows that the plain-text is either 0 or 1, the challenge is for him to tell correctly whether it is 0 or 1.
- Similar in formal verification (with perfect cryptography): static equivalence of frames.

Alpha-Beta Privacy

- Novel approach to formulating privacy:
  - ★ Specify by a logical formula α what information the intruder is allowed to know, e.g. election result:

$$\alpha \equiv v_1, \ldots, v_n \in \{0,1\} \land \sum_{i=0}^{n} v_i = 42$$

  - ★ Specify by a logical formula β what the intruder actually knows, e.g., observed cryptographic messages, what he knows about their structure, etc.
  - ★ Privacy: the intruder cannot derive anything from β except what already follows from α.
- Ongoing research effort by Luca Viganò, Sébastien Gondron, Laouen Fernet and myself.
  - ★ We welcome more colleagues to join us e.g. for a MSc thesis!

Example: Basic Authentication Control (BAC)

RFID passport protocol BAC:

$$
\begin{array}{rl}
\mathit{Tag} \to \mathit{Reader}: & n_T \\
\mathit{Reader} \to \mathit{Tag}: & \{\!|n_R, n_T, k_R|\!\}_{sk_e(\mathit{Tag})}, \\
& \mathrm{mac}(sk_m(\mathit{Tag}), \{\!|n_R, n_T, k_R|\!\}_{sk_e(\mathit{Tag})}) \\
\mathit{Tag} \to \mathit{Reader}: & \mathrm{msg}, \mathrm{mac}(\mathrm{sk_m}(\mathrm{Tag}), \mathrm{msg})
\end{array}
$$

French implementation of the Tag:

$$
\begin{array}{rcl}
P_{\mathrm{Tag}}(\mathit{Tag}) & := & \nu n_T.\mathrm{send}(n_T).\mathrm{receive}(x). \\
& & \mathrm{let}\ (x_1, x_2) = x \\
& & \mathrm{if}\ \mathrm{mac}(sk_m(\mathit{Tag}), x_1) = x_2 \\
& & \mathrm{then\ let}\ (y_1, y_2, y_3) = \mathrm{decrypt}(sk_e(\mathit{Tag}), x_1) \\
& & \qquad \mathrm{if}\ y_2 = n_T \\
& & \qquad \mathrm{then\ send}(\mathrm{msg}, \mathrm{mac}(sk_m(\mathit{Tag}), \mathrm{msg})) \\
& & \qquad \mathrm{else\ send}(\mathrm{errorNonce}) \\
& & \mathrm{else\ send}(\mathrm{errorMac})
\end{array}
$$

Unlinkability goal: you cannot tell if two given sessions where done by the same tag or by different tags. Thus, you cannot link actions.

Formalizing Unlinkability

- Let Tag be the set of all tags (finite).
- Whenever a tag starts a session:
  - Pick a new variable z (that did not occur before) to represent the name of the tag.
  - Release in α the information that $z \in Tag$.
- The intruder is thus allowed to know that it is a Tag...
- After a number of sessions, the intruder thus knows

$$\alpha = z_1 \in Tag \wedge z_2 \in Tag \wedge \ldots \wedge z_n \in Tag$$

Formalizing Unlinkability

- Let *Tag* be the set of all tags (finite).
- Whenever a tag starts a session:
  - ★ Pick a new variable z (that did not occur before) to represent the name of the tag.
  - ★ Release in α the information that $z \in Tag$.
- The intruder is thus allowed to know that it is a *Tag*...
- After a number of sessions, the intruder thus knows

$$\alpha = z_1 \in Tag \wedge z_2 \in Tag \wedge \ldots \wedge z_n \in Tag$$

- It is thus a privacy violation, if the intruder ...
  - ★ can find out the precise identity of any tag, e.g. $z_3 = tag_{42}$;
  - ★ can narrow down the identity of any tag, e.g. $z_3 \in \{tag_{42}, tag_{112}, tag_{1001}\}$;
  - ★ can say that two tags are the same, e.g., $z_3 = z_5$;
  - ★ can say that two tags are different, e.g., $z_3 \neq z_5$;

An Attack

$$\begin{array}{rl} z_1 \to \mathit{Reader}: & n_T \\ \mathit{Reader} \to z_1: & \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\ z_1 \to \mathit{Reader}: & \mathrm{msg}, \mathrm{mac}(\mathrm{sk_m}(z_1), \mathrm{msg}) \end{array}$$

$$\begin{array}{rl} z_2 \to {\color{red} i(\mathit{Reader})}: & n'_T \\ {\color{red} i(\mathit{Reader})} \to z_2: & \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\ z_2 \to {\color{red} i(\mathit{Reader})}: & {\color{red} \mathrm{errorMac}} \end{array}$$

$$\begin{array}{rcl} P_{\mathrm{Tag}}(\mathit{Tag}) & := & \nu n_T.\mathrm{send}(n_T).\mathrm{receive}(x). \\ & & \mathrm{let}\ (x_1, x_2) = x \\ & & \mathrm{if}\ \mathrm{mac}(sk_m(\mathit{Tag}), x_1) = x_2 \\ & & \mathrm{then\ let}\ (y_1, y_2, y_3) = \mathrm{decrypt}(sk_e(\mathit{Tag}), x_1) \\ & & \qquad \mathrm{if}\ y_2 = n_T \\ & & \qquad \mathrm{then\ send}(\mathrm{msg}, \mathrm{mac}(sk_m(\mathit{Tag}), \mathrm{msg})) \\ & & \qquad \mathrm{else\ send}(\mathrm{errorNonce}) \\ & & \mathrm{else\ send}(\mathrm{errorMac}) \end{array}$$

An Attack

$$
\begin{aligned}
z_1 \to Reader &: \quad n_T \\
Reader \to z_1 &: \quad \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\
z_1 \to Reader &: \quad \mathrm{msg}, \mathrm{mac}(\mathrm{sk_m}(z_1), \mathrm{msg})
\end{aligned}
$$

$$
\begin{aligned}
z_2 \to i(Reader) &: \quad n'_T \\
i(Reader) \to z_2 &: \quad \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\
z_2 \to i(Reader) &: \quad \mathrm{errorMac}
\end{aligned}
$$

$$
\begin{aligned}
P_{\mathrm{Tag}}(Tag) \quad := \quad & \nu n_T.\mathrm{send}(n_T).\mathrm{receive}(x). \\
& \mathrm{let}\ (x_1, x_2) = x \\
& \mathrm{if}\ \mathrm{mac}(sk_m(Tag), x_1) = x_2 \\
& \mathrm{then\ let}\ (y_1, y_2, y_3) = \mathrm{decrypt}(sk_e(Tag), x_1) \\
& \qquad \mathrm{if}\ y_2 = n_T \\
& \qquad \mathrm{then\ send}(\mathrm{msg}, \mathrm{mac}(sk_m(Tag), \mathrm{msg})) \\
& \qquad \mathrm{else\ send}(\mathrm{errorNonce}) \\
& \mathrm{else\ send}(\mathrm{errorMac})
\end{aligned}
$$

- If observe errorMac: the mac check must have failed: $z_1 \neq z_2$.

An Attack

$$\begin{aligned}
z_1 \to \mathit{Reader} :&\quad n_T \\
\mathit{Reader} \to z_1 :&\quad \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\
z_1 \to \mathit{Reader} :&\quad \mathrm{msg}, \mathrm{mac}(\mathrm{sk_m}(z_1), \mathrm{msg})
\end{aligned}$$

$$\begin{aligned}
z_2 \to i(\mathit{Reader}) :&\quad n'_T \\
i(\mathit{Reader}) \to z_2 :&\quad \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}, \mathrm{mac}(sk_m(z_1), \{\!|n_R, n_T, k_R|\!\}_{sk_e(z_1)}) \\
z_2 \to i(\mathit{Reader}) :&\quad \mathrm{errorMac}
\end{aligned}$$

$$\begin{aligned}
P_{\mathrm{Tag}}(\mathit{Tag}) \quad := \quad & \nu n_T.\mathrm{send}(n_T).\mathrm{receive}(x). \\
& \mathrm{let}\ (x_1, x_2) = x \\
& \mathrm{if}\ \mathrm{mac}(sk_m(\mathit{Tag}), x_1) = x_2 \\
& \mathrm{then\ let}\ (y_1, y_2, y_3) = \mathrm{decrypt}(sk_e(\mathit{Tag}), x_1) \\
& \qquad \mathrm{if}\ y_2 = n_T \\
& \qquad \mathrm{then\ send}(\mathrm{msg}, \mathrm{mac}(sk_m(\mathit{Tag}), \mathrm{msg})) \\
& \qquad \mathrm{else\ send}(\mathrm{errorNonce}) \\
& \mathrm{else\ send}(\mathrm{errorMac})
\end{aligned}$$

- If observe errorMac: the mac check must have failed: $z_1 \neq z_2$.
- If observe errorNonce: the mac check had worked: $z_1 = z_2$.

The British Implementation

$$
\begin{array}{lcl}
P_{\mathsf{Tag}}(Tag) & := & \nu n_T.\mathsf{send}(n_T).\mathsf{receive}(x). \\
& & \mathsf{let}\ (x_1, x_2) = x \\
& & \mathsf{if}\ \mathsf{mac}(sk_m(Tag), x_1) = x_2 \\
& & \mathsf{then}\ \mathsf{let}\ (y_1, y_2, y_3) = \mathsf{decrypt}(sk_e(Tag), x_1) \\
& & \qquad \mathsf{if}\ y_2 = n_T \\
& & \qquad \mathsf{then}\ \mathsf{send}(\mathsf{msg}, \mathsf{mac}(sk_m(Tag), \mathsf{msg})) \\
& & \qquad \mathsf{else}\ \mathsf{send}(\mathsf{error}) \\
& & \mathsf{else}\ \mathsf{send}(\mathsf{error})
\end{array}
$$

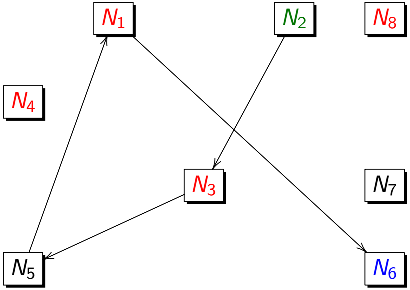
The British Implementation

$$
\begin{aligned}
P_{\mathsf{Tag}}(Tag) \quad := \quad & \nu n_T.\mathsf{send}(n_T).\mathsf{receive}(x). \\
& \mathsf{let}\ (x_1, x_2) = x \\
& \mathsf{if}\ \mathsf{mac}(sk_m(Tag), x_1) = x_2 \\
& \mathsf{then\ let}\ (y_1, y_2, y_3) = \mathsf{decrypt}(sk_e(Tag), x_1) \\
& \qquad \mathsf{if}\ y_2 = n_T \\
& \qquad \mathsf{then\ send}(\mathsf{msg}, \mathsf{mac}(sk_m(Tag), \mathsf{msg})) \\
& \qquad \mathsf{else\ send}(\mathsf{error}) \\
& \mathsf{else\ send}(\mathsf{error})
\end{aligned}
$$

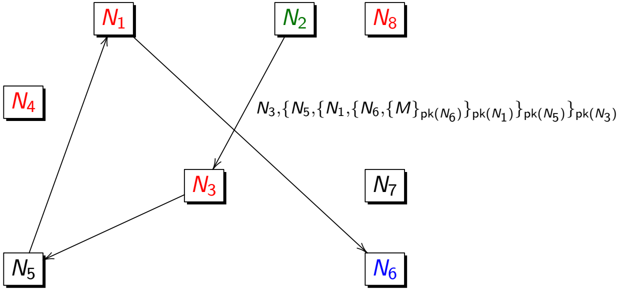
- Unlinkable! (as can be shown)

Outline

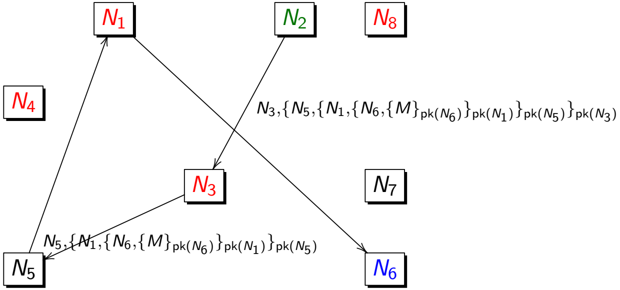
Mixes/Onion Routing



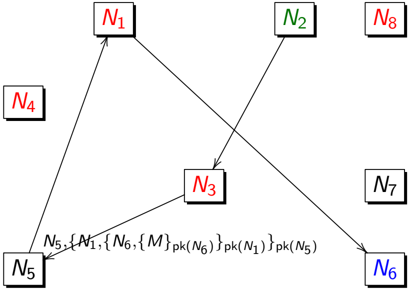
Mixes/Onion Routing



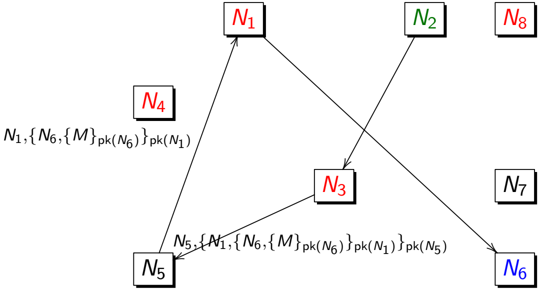
Mixes/Onion Routing



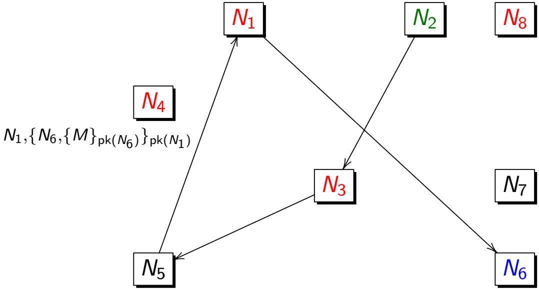
Mixes/Onion Routing



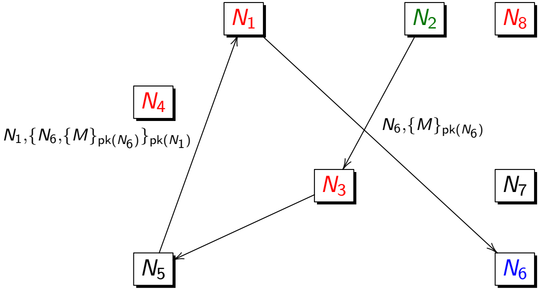
Mixes/Onion Routing



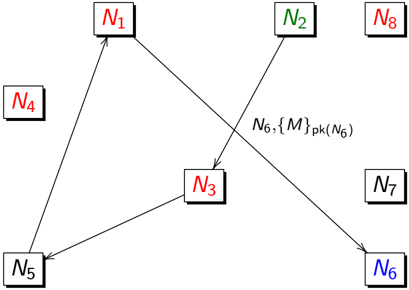
Mixes/Onion Routing



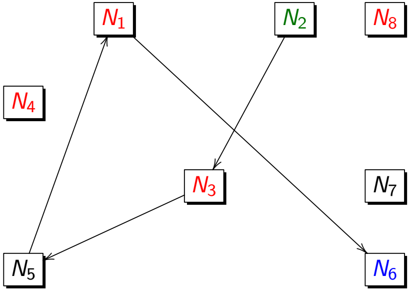
Mixes/Onion Routing



Mixes/Onion Routing

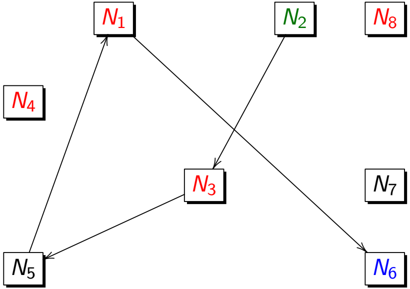


Mixes/Onion Routing



- Each node knows only who is the previous/next on the chain
- Only the destination learns the message M
- Nobody knows who is source except the source itself
- Nobody knows who is the destination (except source and destination)
- Attacker model: can see all exchanged messages and may control some nodes (may include destination)

Mixes/Onion Routing: Assumptions



- Sender knows the true public key of all nodes.
- At least one node on the path is not controlled by the attacker
- Traffic is evenly distributed between all nodes
- Replay of messages is prevented
- Length of an encrypted message does not reveal the number of encryption layers.
- Note also: if destination is not part of mix network, cleartext messages are transmitted on the last leg.

Outline

Idea

Zero-knowledge proofs

In zero-knowledge proofs we can usually specify a statement that is being proved.

- Definitely, that statement is revealed to the verifier
- The verifier (or others) should not learn anything else
- Everybody can draw conclusions from everything they learned

Zero-Knowledge Protocols

Scenario

- A Prover Peggy and a Verifier Victor
- Peggy has a secret and wants to convince Victor of that fact without telling the secret.

- Example Schnorr protocol:
  - ★ Victor knows a public value $Z \in \mathbb{Z}_p^\star$.
  - ★ Peggy wants to prove that she knows $X \in \mathbb{Z}_p^\star$ with $Z \equiv_p g^X$.
- Why? What's the point of proving this?

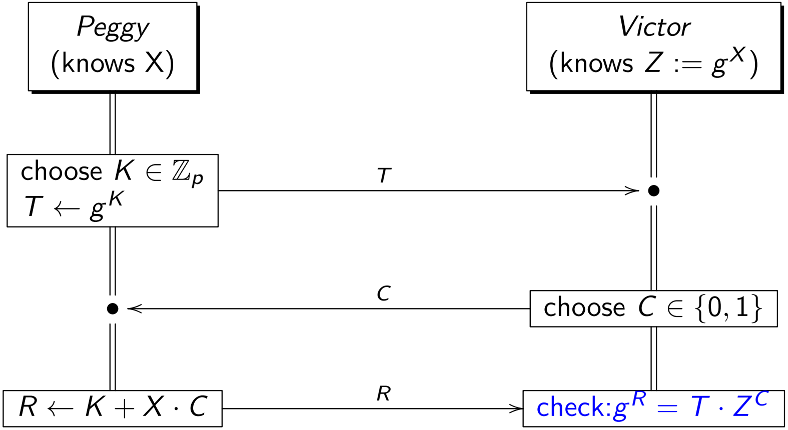
Zero-Knowledge Protocols

Scenario

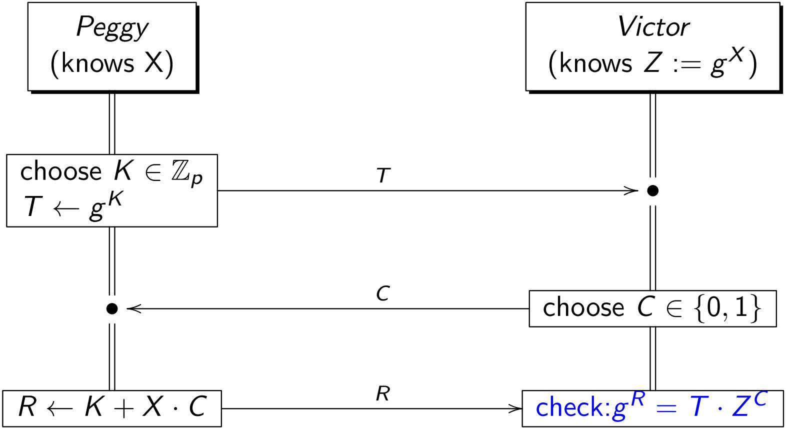
- A Prover Peggy and a Verifier Victor
- Peggy has a secret and wants to convince Victor of that fact without telling the secret.

- Example Schnorr protocol:
  - Victor knows a public value $Z \in \mathbb{Z}_p^\star$.
  - Peggy wants to prove that she knows $X \in \mathbb{Z}_p^\star$ with $Z \equiv_p g^X$.
- Why? What's the point of proving this?
  - Peggy can authenticate this way
  - Anonymous credential systems ("Private authentication")
  - Voting protocols like Helios/Belenios
  - Alternatives to Proof-of-Work in blockchain implementations.

Schnorr: the Protocol



Schnorr: the Protocol



If Peggy worked correctly:

$$g^R = g^{K+X\cdot C} = g^K \cdot g^{X \cdot C} = T \cdot Z^C$$

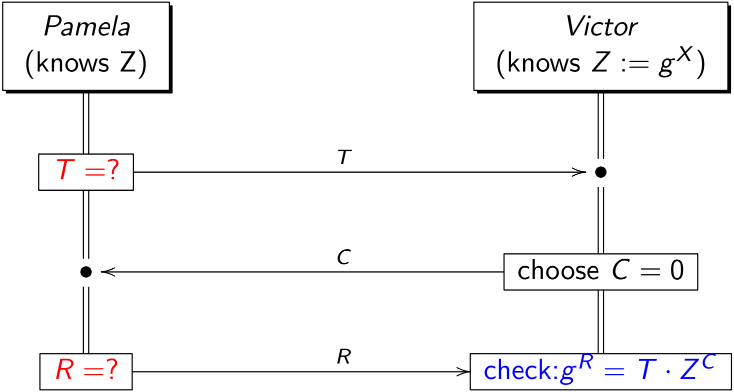
Schnorr: Trying to Cheat

Pamela does not know the secret, but tries to prove its knowledge.
Pamela guesses that Victor is going to choose $C = 0$

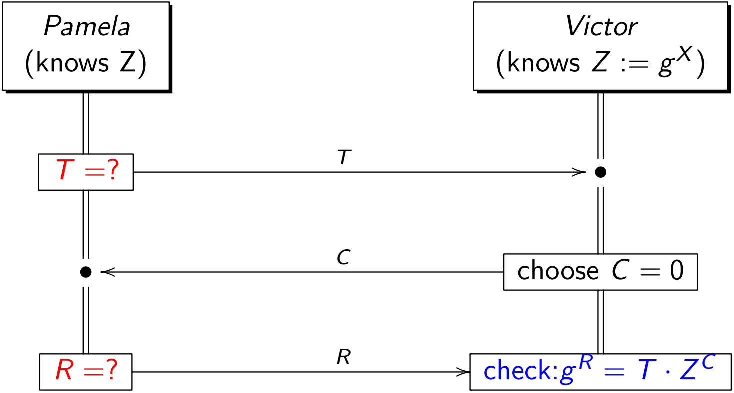
Schnorr: Trying to Cheat

Pamela does not know the secret, but tries to prove its knowledge.

Pamela guesses that Victor is going to choose $C = 0$



Pamela does not know the secret, but tries to prove its knowledge.
Pamela guesses that Victor is going to choose $C = 0$



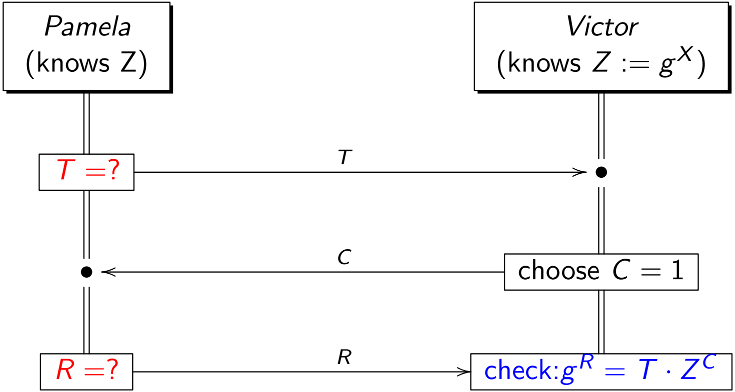
Strategy for Pamela: Choose R randomly and set $T = g^R$.

Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose $C = 1$

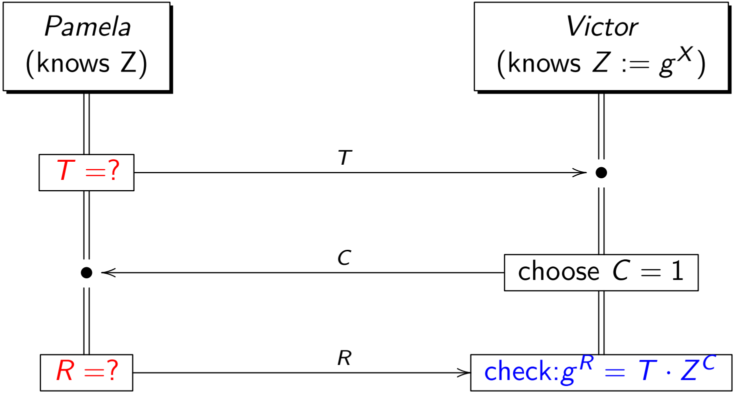
Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose $C = 1$



Schnorr: Trying to Cheat

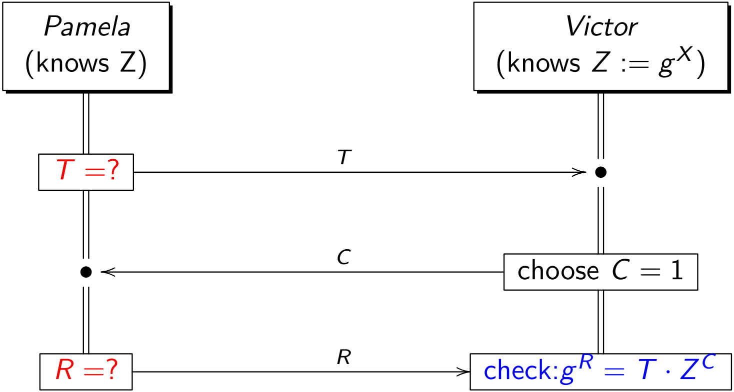
Pamela guesses that Victor is going to choose $C = 1$



Find numbers T and R such that $g^R = T \cdot Z$.

Schnorr: Trying to Cheat

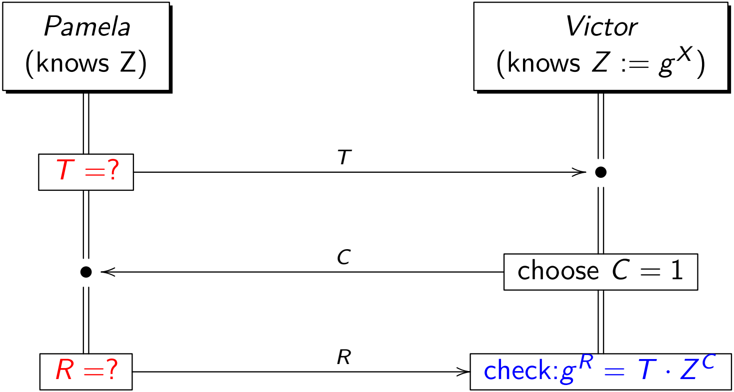
Pamela guesses that Victor is going to choose $C = 1$



Find numbers T and R such that $g^R = T \cdot Z$.
Hint: division modulo p is also easy!

Schnorr: Trying to Cheat

Pamela guesses that Victor is going to choose $C = 1$



Find numbers T and R such that $g^R = T \cdot Z$.
Hint: division modulo p is also easy!
Strategy for Pamela: Choose R randomly, set $T = \frac{g^R}{Z}$.

Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if $C = 0$:

Choose R randomly, set $T = g^R$.

Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if $C = 0$:

Choose R randomly, set $T = g^R$.

- Pamela has a strategy to cheat if $C = 1$:

Choose R randomly, set $T = \frac{g^R}{Z}$

Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if $C = 0$:

Choose R randomly, set $T = g^R$.

- Pamela has a strategy to cheat if $C = 1$:

Choose R randomly, set $T = \frac{g^R}{Z}$

- Since T must be sent before C is known, she can prepare only for one possibility
  - ★ Chance of $\frac{1}{2}$ to cheat (if $C \in \{0, 1\}$ uniformly chosen)

Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if $C = 0$:

Choose R randomly, set $T = g^R$.

- Pamela has a strategy to cheat if $C = 1$:

Choose R randomly, set $T = \frac{g^R}{Z}$

- Since T must be sent before C is known, she can prepare only for one possibility
  - ★ Chance of $\frac{1}{2}$ to cheat (if $C \in \{0, 1\}$ uniformly chosen)
- Claim: Pamela cannot do better (to prove...)

Schnorr: Trying to Cheat

- Pamela has a strategy to cheat if $C = 0$:

Choose R randomly, set $T = g^R$.

- Pamela has a strategy to cheat if $C = 1$:

Choose R randomly, set $T = \frac{g^R}{Z}$

- Since T must be sent before C is known, she can prepare only for one possibility
  - ★ Chance of $\frac{1}{2}$ to cheat (if $C \in \{0, 1\}$ uniformly chosen)
- Claim: Pamela cannot do better (to prove...)
- If Victor accepts only after n successful rounds of the protocol, the chance to cheat is only 2^{-n}.

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.
- Then she has values T, R_0, and R_1 such that
  - ★ (If Victor chooses $C = 0$:) $g^{R_0} = T$
  - ★ (If Victor chooses $C = 1$:) $g^{R_1} = T \cdot Z$

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.
- Then she has values T, R_0, and R_1 such that
  - ★ (If Victor chooses $C = 0$:) $g^{R_0} = T$
  - ★ (If Victor chooses $C = 1$:) $g^{R_1} = T \cdot Z$
- Then she can compute $Q = R_1 - R_0$.

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.
- Then she has values T, R_0, and R_1 such that
  - ★ (If Victor chooses $C = 0$:) $g^{R_0} = T$
  - ★ (If Victor chooses $C = 1$:) $g^{R_1} = T \cdot Z$
- Then she can compute $Q = R_1 - R_0$.
- It holds that $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$.

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.
- Then she has values T, R_0, and R_1 such that
  - ★ (If Victor chooses $C = 0$:) $g^{R_0} = T$
  - ★ (If Victor chooses $C = 1$:) $g^{R_1} = T \cdot Z$
- Then she can compute $Q = R_1 - R_0$.
- It holds that $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$.
- So with $Q = X$ she indeed knows the discrete logarithm of Z.

Schnorr: Trying to Cheat

Claim

Pamela has no strategy to cheat for unpredictable C.

Proof sketch

- Suppose Pamela has a strategy that works both for $C = 0$ and for $C = 1$.
- Then she has values T, R_0, and R_1 such that
  - ★ (If Victor chooses $C = 0$:) $g^{R_0} = T$
  - ★ (If Victor chooses $C = 1$:) $g^{R_1} = T \cdot Z$
- Then she can compute $Q = R_1 - R_0$.
- It holds that $g^Q = g^{R_1 - R_0} = \frac{g^{R_1}}{g^{R_0}} = \frac{T \cdot Z}{T} = Z$.
- So with $Q = X$ she indeed knows the discrete logarithm of Z.
- So it is not cheating!

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$$\begin{array}{ccc} T_1 & C_1 & R_1 \\ T_2 & C_2 & R_2 \\ & \dots & \\ T_n & C_n & R_n \end{array}$$

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$$\begin{array}{ccc} T_1 & C_1 & R_1 \\ T_2 & C_2 & R_2 \\ & \dots & \\ T_n & C_n & R_n \end{array}$$

Create a fake-transcript using random challenges C_i'

$$\begin{array}{c} C_1' \\ C_2' \\ \dots \\ C_n' \end{array}$$

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

Consider a transcript of an exchange between Peggy and Victor. (Assume Victor chooses challenges randomly.)

$$\begin{array}{ccc} T_1 & C_1 & R_1 \\ T_2 & C_2 & R_2 \\ & \dots & \\ T_n & C_n & R_n \end{array}$$

Create a fake-transcript using random challenges C_i' and then filling the T_i' and R_i' according to the cheat strategies of Pamela for known challenges:

$$\begin{array}{ccc} T_1' & C_1' & R_1' \\ T_2' & C_2' & R_2' \\ & \dots & \\ T_n' & C_n' & R_n' \end{array}$$

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?
- No: statically indistinguishable. (Not even distinguishable with unbounded computing resources!)

Schnorr: Curious Victor

Victor would like to learn the secret X...

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Proof sketch

- Give the two transcripts to a third party. Can one tell the real from the fake?
- No: statically indistinguishable. (Not even distinguishable with unbounded computing resources!)
- So, whatever information Victor is getting out of the transcript, he may have created that himself without Peggy!

The Mafia-owned Restaurant

Claim (Shamir)

I can go to a Mafia-owned restaurant and pay with my zero-knowledge proof-based credit card frequently, and yet the mobsters cannot impersonate me.

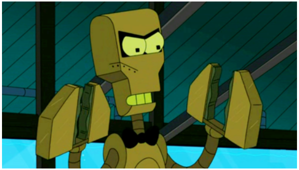
The Mafia-owned Restaurant

Claim (Shamir)

I can go to a Mafia-owned restaurant and pay with my zero-knowledge proof-based credit card frequently, and yet the mobsters cannot impersonate me.

They can, if they do it online.

The Mafia Attack



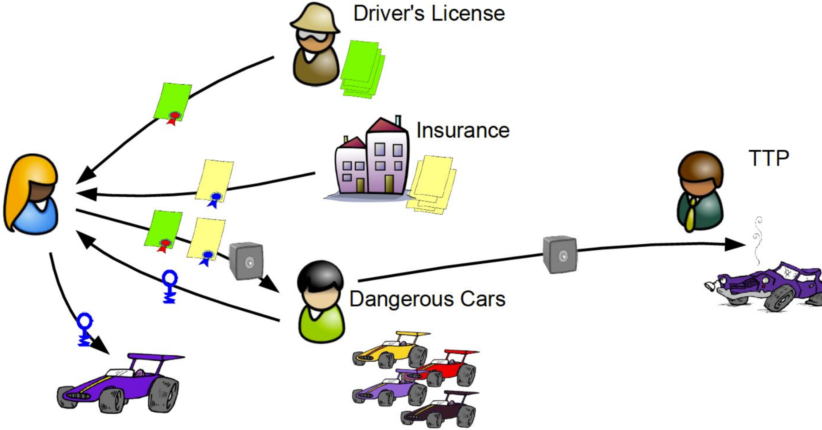
Classical Man in the middle attack: forward all messages!

Countermeasures

- Distance bounding protocols.
- Link the intended verifier to the statement being proved.

IBM's Idemix

A privacy friendly credential system



IBM's Idemix

- Credentials: list of attribute-value pairs, signed by an issuer.
  sign(copenhagen-commune; name="Johanna Gossner", birthdate=14/12/73, kind="Drivers License", class="3",...)
- Signatures are special: Camenisch-Lysyanskaya. Allows for special zero-knowledge proofs
  - ★ Proving possession of credential without transmitting it
  - ★ Revealing only some attributes (e.g. kind and class)
  - ★ Proving properties about attributes (e.g. birthdate<9/10/2000)
  - ★ Proving relations between credentials (e.g. passport on same name)
  - ★ Signing a statement (e.g. "I confirm this electronic order...")

IBM's Idemix

- Credentials: list of attribute-value pairs, signed by an issuer.
  sign(copenhagen-commune; name="Johanna Gossner", birthdate=14/12/73, kind="Drivers License", class="3",...)
- Signatures are special: Camenisch-Lysyanskaya. Allows for special zero-knowledge proofs
  - ★ Proving possession of credential without transmitting it
  - ★ Revealing only some attributes (e.g. kind and class)
  - ★ Proving properties about attributes (e.g. birthdate<9/10/2000)
  - ★ Proving relations between credentials (e.g. passport on same name)
  - ★ Signing a statement (e.g. "I confirm this electronic order...")
- Privacy: A verifier cannot find out more than what was proved.
  - ★ E.g. different transactions are unlikable (unless linked by prover).
  - ★ Holds even if the issuer is dishonest and collaborates with a dishonest verifier.

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

- Can be achieved using escrow with verifiable encryptions

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

- Can be achieved using escrow with verifiable encryptions
- Provers have to encrypt their real names with the public key of an authority.

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

- Can be achieved using escrow with verifiable encryptions
- Provers have to encrypt their real names with the public key of an authority.
- Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

- Can be achieved using escrow with verifiable encryptions
- Provers have to encrypt their real names with the public key of an authority.
- Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).
- Then the authority can revoke their privacy when needed.

Idemix: Privacy with Accountability?

How can we ensure that we

- protect the privacy of the innocent
- but not the privacy of the guilty?

- Can be achieved using escrow with verifiable encryptions
- Provers have to encrypt their real names with the public key of an authority.
- Provers have to prove (in zero-knowledge) that they have encrypted the real name (and it is correctly encrypted).
- Then the authority can revoke their privacy when needed.
- This should be tied to a legal system, e.g. the privacy revocation happens only on court order.