Formal verification of cryptographic protocols

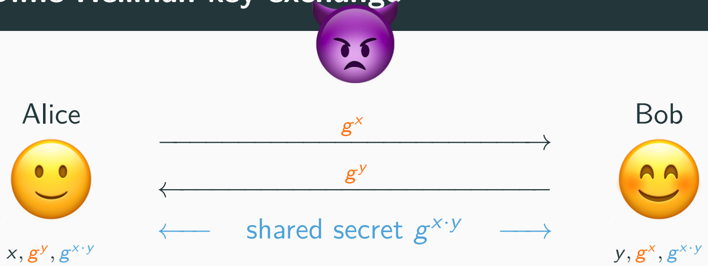
David Baelde, Joseph Lallemand

1 July 2025

Introduction

- Cryptographic protocols are used to secure communications over insecure networks
- All kinds of applications
  e.g. Web (HTTPS/TLS), Instant messaging (Signal), Wi-Fi (WPA),
  Credit card payment (EME), 4G/5G (AKA)...
- Very often they are flawed, leading to attacks
- We want to analyse protocols to formally prove the absence of vulnerabilities

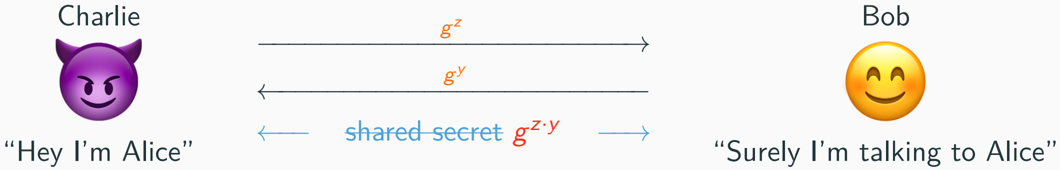
Example: the Diffie-Hellman key exchange



- Alice and Bob establish a shared secret $g^{x \cdot y}$
- Relies on the Diffie-Hellman assumption on the group:

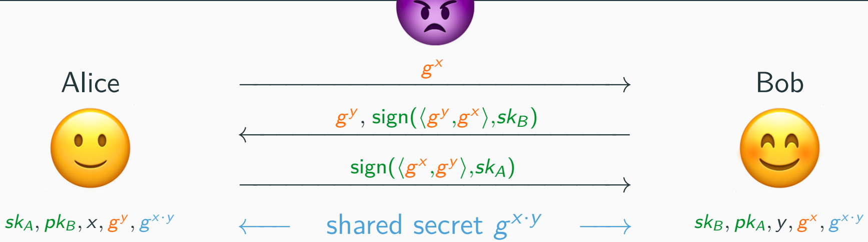
It is hard to compute $g^{x \cdot y}$ knowing only g^x and g^y.

The Need for Authentication



- That’s the general idea, but it’s not enough
- No authentication! Charlie could impersonate Alice.
- Bob computes $g^{z \cdot y}$, which is not secret – Charlie knows it.

The Signed Diffie-Hellman key exchange



- Alice and Bob sign the two values g^x, g^y
- They authenticate each other, and agree on $g^{x \cdot y}$.
- For that, signatures need to be unforgeable:

It is hard to forge a signature sign(m, sk) without knowing the key sk.

Process notation

We often use a process notation inspired by the π-calculus.

$$
\begin{aligned}
& P_{Alice}(sk_A, pk_B) = \\
& \qquad \text{new } x; \\
& \qquad \text{out}(g^x); \\
& \qquad \text{in}(m); \\
& \qquad \text{let } \langle Y', s \rangle = m \text{ in} \\
& \qquad \text{if verify}(\langle Y', g^x \rangle, s, pk_B) \text{ then} \\
& \qquad \quad \text{out}(\text{sign}(\langle g^x, Y' \rangle, sk_A)).
\end{aligned}
$$

$$
\begin{aligned}
& P_{Bob}(sk_B, pk_A) = \\
& \qquad \text{in}(X'); \\
& \qquad \text{new } y; \\
& \qquad \text{out}(\langle g^y, \text{sign}(\langle g^y, X' \rangle, sk_B) \rangle); \\
& \qquad \text{in}(s); \\
& \qquad \text{if verify}(\langle X', g^y \rangle, s, pk_A) \text{ then} \\
& \qquad \quad \text{out}(\text{👍}).
\end{aligned}
$$

$$
\begin{aligned}
P_{DH} = \quad & \text{new } sk_A;\ \text{new } sk_B; \\
& \text{out}(\text{pk}(sk_A));\ \text{out}(\text{pk}(sk_B)); \\
& (P_{Alice}(sk_A, \text{pk}(sk_B)) \mid P_{Bob}(sk_B, \text{pk}(sk_A)))
\end{aligned}
$$

MITM attack & the actual signed Diffie-Hellman protocol

- In the end, Bob incorrectly believes he is talking to Alice 😭

MITM attack & the actual signed Diffie-Hellman protocol

Alice | Charlie | Bob

$$\text{Alice} \xrightarrow{g^x} \text{Charlie} \xrightarrow{g^x} \text{Bob}$$

$$\text{Alice} \xleftarrow{g^y,\ \mathrm{sign}(\langle A, g^y, g^x\rangle, sk_C)} \text{Charlie} \xleftarrow{g^y,\ \mathrm{sign}(\langle A, g^y, g^x\rangle, sk_B)} \text{Bob}$$

$$\text{Alice} \xrightarrow{\mathrm{sign}(\langle C, g^x, g^y\rangle, sk_A)} \text{Charlie} \xrightarrow{\mathrm{sign}(\langle C, g^x, g^y\rangle, sk_A) ✗} \text{Bob}$$

"Hi Charlie!" "Hey I'm Charlie!" "Hey I'm Alice!" "Wait that's not Alice!"

- In the end, Bob incorrectly believes he is talking to Alice 😭
- Fix: adding the identities of A and B in the signatures.

Formal analysis of protocols

- Our goal: prove that there are no such attacks.
- First, we need to construct formal models of
  - the protocol we study
  - the attacker we want to defend against
  - the properties the protocol should ensure
- Then prove that, in that model, no attacker can break the properties 😎

Formal analysis of protocols

- Our goal: prove that there are no such attacks.
- First, we need to construct formal models of
  - the protocol we study
  - the attacker we want to defend against
  - the properties the protocol should ensure
- Then prove that, in that model, no attacker can break the properties 😎

Just one problem: proofs tend to be difficult and painful and full of errors 😒

We want **mechanised tools** to help us with that.

Security properties

Confidentiality

Confidentiality property

Some data can only be learned by authorised participants,
but remains secret to an attacker.

For instance:

- A key that has been exchanged
- A password
- A message
- A movie

Authentication

Authentication property

An agent can be sure of the identity of the entity they are talking to.

For instance:

- A service provider authenticates a user
- A 4G operator authenticates a phone
- A web browser authenticates a server

Privacy properties (examples)

Anonymity

An attacker cannot find out which agent is executing the protocol.

Unlinkability

An attacker cannot link multiple protocol sessions of the same agent
i.e. find out whether two sessions belong to the same agent.

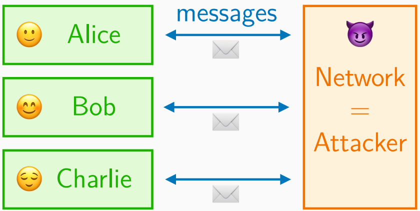
Vote privacy

An attacker cannot find out which voter voted for which candidate.

Models and tools

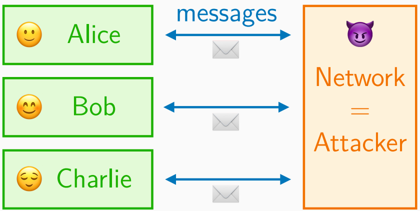
Attacker models

- We need a model of the attacker we want to defend against
- Basically: an attacker who controls the network



Attacker models

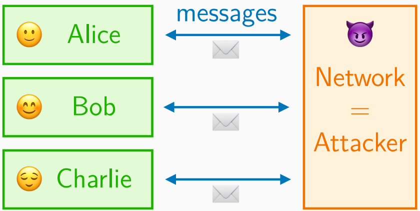
- We need a model of the attacker we want to defend against
- Basically: an attacker who controls the network



- What about the attacker's computing power?

Attacker models

- We need a model of the attacker we want to defend against
- Basically: an attacker who controls the network



- What about the attacker's computing power?
- Two kinds of models: Computational and Symbolic models

Symbolic model / Dolev-Yao model

- Very abstract representation of everything
- Cryptographic primitives are assumed to be perfect
- Logical frameworks to model protocols and messages
  e.g. state machines, transition systems, rewriting systems, process algebras...
- Attacker has full control of the network, but limited computation capabilities due to strong assumptions on cryptography
- Very good automation 👍, at the cost of somewhat weak guarantees 👎

Symbolic model: tools

ProVerif

Automated tool for protocol verification.

Protocols modelled as π-calculus processes, incomplete procedure (does not always conclude)

Tamarin

Automated/interactive tool for protocol verification.

Protocols modelled as multiset rewriting rules, incomplete procedure (does not always terminate)

Bounded tools: Deepsec, Akiss, . . .

Decision procedures to prove security for bounded numbers of sessions (always terminate and conclude).

Computational Model

Computational model – General ideas

- Attacker and protocol participants are (probabilistic) Turing machines, run in polynomial time w.r.t. the size of keys used
- Precise assumptions on cryptographic primitives, expressed as cryptographic games
  - *e.g.* IND-CCA, EUF-CMA
- Proofs by reduction on the games
- Precise, realistic 👍 but very hard to automate proofs 👎

Computational model – Formal analysis tools

CryptoVerif

Automated procedure to perform cryptographic game transformations.

EasyCrypt

Proof assistant to reason about probabilistic programs,
More geared towards proving cryptographic primitives.

Squirrel 🥰🐿️

Proof assistant to reason about protocols with a more abstract view,
It's amazing $\longrightarrow$ more on that very soon.

Computational model – Security parameter

- Study the probability of an adversary breaking security

Computational model – Security parameter

- Study the probability of an adversary breaking security
- Everything is parametrised by the security parameter η
  i.e. the size of keys and other randomly sampled values (nonces)

Computational model – Security parameter

- Study the probability of an adversary breaking security
- Everything is parametrised by the security parameter η
  i.e. the size of keys and other randomly sampled values (nonces)
- The adversary is a Probabilistic Polynomial-time Turing Machine (PPTM) w.r.t. η
  - Polynomial time: discard brute force attacks
  - Probabilistic: could always guess keys at random, with probability $2^{-\eta}$

Computational model – Security parameter

- Study the probability of an adversary breaking security
- Everything is parametrised by the security parameter η
  i.e. the size of keys and other randomly sampled values (nonces)
- The adversary is a Probabilistic Polynomial-time Turing Machine (PPTM) w.r.t. η
  - Polynomial time: discard brute force attacks
  - Probabilistic: could always guess keys at random, with probability $2^{-\eta}$
- Security can only hold up to negligible probability

A function $f : \mathbb{N} \mapsto \mathbb{R}$ is **negligible**, written $f(n) \in \text{negl}(n)$, if

$$\forall k.\ \exists n_0.\ \forall n \geq n_0.\ f(n) \leq n^{-k}$$

Cryptographic assumptions

- Cryptographic primitives are also poly time algorithms, may be randomised
- The security of a protocol relies on the security of primitives
- Assumptions (at least for us:
  - Correctness assumptions, *e.g.* $\mathrm{verify}(m, \mathrm{sign}(m, sk), pk(sk)) = \top$.
  - Security assumptions, formalised as cryptographic games
- A game is an experiment where an adversary tries to break the primitive in a specific way.
  We assume he only has a negligible advantage ($\approx$ probability of success)

Computational Diffie-Hellman (CDH) assumption

- "It is hard to compute $g^{x \cdot y}$ from g^x, g^y"

Computational Diffie-Hellman (CDH) assumption

- "It is hard to compute $g^{x \cdot y}$ from g^x, g^y"
- Assume an algorithm $\mathsf{gen}_{\mathrm{DH}}$, that produces a cyclic group G, with a generator g.

Computational Diffie-Hellman (CDH) assumption

- "It is hard to compute $g^{x \cdot y}$ from g^x, g^y"
- Assume an algorithm $\mathsf{gen}_{\mathsf{DH}}$, that produces a cyclic group G, with a generator g.

$\mathsf{Exp}^{\mathsf{CDH}}_{\mathcal{A}}(\eta)$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$

$x \leftarrow\$ [\![0, |G| - 1]\!]$

$y \leftarrow\$ [\![0, |G| - 1]\!]$

$z \leftarrow \mathcal{A}(1^\eta, G, g, g^x, g^y)$

return $(z = g^{x \cdot y})$

Computational Diffie-Hellman (CDH) assumption

- "It is hard to compute $g^{x \cdot y}$ from g^x, g^y"
- Assume an algorithm $\mathsf{gen}_{\mathsf{DH}}$, that produces a cyclic group G, with a generator g.
- Advantage:
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CDH}}(\eta) = \mathrm{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CDH}}(\eta) = 1\right]$$
- The CDH assumption is that for any PPTM $\mathcal{A}$,
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{CDH}}(\eta) \in \mathsf{negl}(\eta)$$

$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{CDH}}(\eta)$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$
$x \xleftarrow{\$} [\![0, |G| - 1]\!]$
$y \xleftarrow{\$} [\![0, |G| - 1]\!]$
$z \leftarrow \mathcal{A}(1^\eta, G, g, g^x, g^y)$
return $(z = g^{x \cdot y})$

Existential Unforgeability under Chosen Message Attacks (EUF-CMA)

- "Signatures cannot be forged without knowing *sk*"

Existential Unforgeability under Chosen Message Attacks (EUF-CMA)

- "Signatures cannot be forged without knowing *sk*"
- For a signature scheme $(\mathsf{gen}_{\mathsf{sign}}, \mathsf{sign}, \mathsf{verify})$

Existential Unforgeability under Chosen Message Attacks (EUF-CMA)

$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\eta)$

$pk, sk \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$L \leftarrow []$

$m_0, s_0 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}}(1^\eta, pk)$

if $\mathsf{verify}(m_0, s_0, pk) \ \wedge \ m_0 \notin L$

then return 1

else return 0

$\mathcal{O}_{\mathsf{sign}}(m)$

$s \leftarrow \mathsf{sign}(m, sk)$

$L \leftarrow m :: L$

return s

- "Signatures cannot be forged without knowing sk"
- For a signature scheme $(\mathsf{gen}_{\mathsf{sign}}, \mathsf{sign}, \mathsf{verify})$
- The adversary has access to a signing oracle $\mathcal{O}_{\mathsf{sign}}$

Existential Unforgeability under Chosen Message Attacks (EUF-CMA)

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\eta)}$

$pk, sk \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$L \leftarrow []$

$m_0, s_0 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}}(1^\eta, pk)$

if $\mathsf{verify}(m_0, s_0, pk) \ \wedge\ m_0 \notin L$

then return 1

else return 0

$\underline{\mathcal{O}_{\mathsf{sign}}(m)}$

$s \leftarrow \mathsf{sign}(m, sk)$

$L \leftarrow m :: L$

return s

- "Signatures cannot be forged without knowing sk"
- For a signature scheme $(\mathsf{gen}_{\mathsf{sign}}, \mathsf{sign}, \mathsf{verify})$
- The adversary has access to a signing oracle $\mathcal{O}_{\mathsf{sign}}$
- Advantage:
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\eta) = \mathsf{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\eta) = 1\right]$$
- The EUF-CMA assumption is that for any PPTM $\mathcal{A}$,
$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{EUF-CMA}}(\eta) \in \mathsf{negl}(\eta)$$

Indistinguishability under Chosen Plaintext Attacks (IND-CPA)

- "Ciphertexts hide their contents"

Indistinguishability under Chosen Plaintext Attacks (IND-CPA)

- “Ciphertexts hide their contents”

- For a symmetric encryption scheme $(\mathsf{gen}_{\mathsf{enc}}, \mathsf{enc}, \mathsf{dec})$

Indistinguishability under Chosen Plaintext Attacks (IND-CPA)

- "Ciphertexts hide their contents"
- For a symmetric encryption scheme $(\mathsf{gen}_{\mathsf{enc}}, \mathsf{enc}, \mathsf{dec})$
- The adversary is given an encryption oracle $\mathcal{O}_{\mathsf{enc}}$
- An indistinguishability game

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IND-CPA},\beta}(\eta)}$

$k \leftarrow \mathsf{gen}_{\mathsf{enc}}(1^\eta)$

$m_0, m_1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}}(1^\eta)$

$c_\beta \leftarrow \mathsf{enc}(m_\beta, pk)$

$\beta' \leftarrow \mathcal{A}(1^\eta, c_\beta)$

return β'

$\underline{\mathcal{O}_{\mathsf{enc}}(m)}$

return $\mathsf{enc}(m, k)$

Indistinguishability under Chosen Plaintext Attacks (IND-CPA)

- "Ciphertexts hide their contents"
- For a symmetric encryption scheme $(\mathsf{gen}_{\mathsf{enc}}, \mathsf{enc}, \mathsf{dec})$
- The adversary is given an encryption oracle $\mathcal{O}_{\mathsf{enc}}$
- An indistinguishability game
- Advantage:
  $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND-CPA}}(\eta) =$
  $\left|\mathsf{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IND-CPA},0}(\eta) = 1\right] - \mathsf{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IND-CPA},1}(\eta) = 1\right]\right|$
- The IND-CPA assumption is that for any PPTM $\mathcal{A}$,
  $$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IND-CPA}}(\eta) \in \mathsf{negl}(\eta)$$

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IND-CPA},\beta}(\eta)}$

$k \leftarrow \mathsf{gen}_{\mathsf{enc}}(1^\eta)$

$m_0, m_1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}}(1^\eta)$

$c_\beta \leftarrow \mathsf{enc}(m_\beta, pk)$

$\beta' \leftarrow \mathcal{A}(1^\eta, c_\beta)$

return β'

$\underline{\mathcal{O}_{\mathsf{enc}}(m)}$

return $\mathsf{enc}(m, k)$

Security of protocols as cryptographic games

- Games are also used to specify security properties of protocols
- As for primitives, an adversary tries to break the security of the protocol in a specific way, *e.g.* learn a secret, ...
- The adversary has access to a set of oracles, to interact with the protocol

The Signed Diffie-Hellman protocol as a set of oracles

Alice $\xrightarrow{g^x}$ Bob

Alice $\xleftarrow{g^y,\ \mathsf{sign}(\langle A, g^y, g^x\rangle, sk_B)}$ Bob

Alice $\xrightarrow{\mathsf{sign}(\langle B, g^x, g^y\rangle, sk_A)}$ Bob

$$\Longrightarrow \quad \mathcal{O} = \{\mathcal{O}_{\mathsf{alice1}}, \mathcal{O}_{\mathsf{alice2}}, \mathcal{O}_{\mathsf{bob1}}, \mathcal{O}_{\mathsf{bob2}}\}$$

$\mathcal{O}_{\mathsf{alice1}}()$

$x \leftarrow\!\!\$ \ [\![0, |G| - 1]\!]$

return g^x

$\mathcal{O}_{\mathsf{bob1}}(m)$

$X' \leftarrow m$

$y \leftarrow\!\!\$ \ [\![0, |G| - 1]\!]$

return $\langle g^y, \mathsf{sign}(\langle A, g^y, X'\rangle, sk_B)\rangle$

$\mathcal{O}_{\mathsf{alice2}}(m, s)$

$Y' \leftarrow m$

if $\mathsf{verify}(\langle A, Y', g^x\rangle, s, pk_B)$ **then**

  return $\mathsf{sign}(\langle B, g^x, Y'\rangle, sk_A)$

$\mathcal{O}_{\mathsf{bob2}}(s)$

---

if $\mathsf{verify}(\langle B, X', g^y\rangle, s, pk_A)$ **then**

  $ok \leftarrow 1$

Secrecy of the Diffie-Hellman exchange

$$
\begin{array}{l}
\underline{\mathsf{Exp}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta)} \\
G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta) \\
pk_A, sk_A \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta) \\
pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta) \\
x, y, X', Y', ok \leftarrow \perp \\
z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B) \\
\mathbf{return}\ (z = g^{x \cdot y})
\end{array}
$$

- "$g^{x \cdot y}$ remains secret"

Secrecy of the Diffie-Hellman exchange

$\underline{\mathsf{Exp}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta)}$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$

$pk_A, sk_A \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$x, y, X', Y', ok \leftarrow \perp$

$z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B)$

return $(z = g^{x \cdot y})$

- "$g^{x \cdot y}$ remains secret"
- The adversary is only allowed one call to each oracle: models a single session

Secrecy of the Diffie-Hellman exchange

$\underline{\mathsf{Exp}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta)}$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$

$pk_A, sk_A \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$x, y, X', Y', ok \leftarrow \perp$

$z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B)$

return $(z = g^{x \cdot y})$

- "$g^{x\cdot y}$ remains secret"
- The adversary is only allowed one call to each oracle: models a single session
- Advantage:
  $$\mathsf{Adv}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta) = \mathrm{P}\left[\mathsf{Exp}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta) = 1\right]$$
- $g^{x\cdot y}$ is secret if for any PPTM $\mathcal{A}$,
  $$\mathsf{Adv}^{\mathsf{secrecy}}_{\mathcal{A}}(\eta) \in \mathsf{negl}(\eta)$$

Secrecy of the Diffie-Hellman exchange

$\underline{\mathrm{Exp}_{\mathcal{A}}^{\mathrm{secrecy}}(\eta)}$

$G, g \leftarrow \mathrm{gen}_{\mathrm{DH}}(1^\eta)$

$pk_A, sk_A \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta)$

$pk_B, sk_B \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta)$

$x, y, X', Y', ok \leftarrow \perp$

$z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B)$

return $(z = g^{x \cdot y})$

- "$g^{x \cdot y}$ remains secret"
- The adversary is only allowed one call to each oracle: models a single session
- Advantage:
  $$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{secrecy}}(\eta) = \mathrm{P}\left[\mathrm{Exp}_{\mathcal{A}}^{\mathrm{secrecy}}(\eta) = 1\right]$$
- $g^{x \cdot y}$ is secret if for any PPTM $\mathcal{A}$,
  $$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{secrecy}}(\eta) \in \mathrm{negl}(\eta)$$
- We could also (more interestingly) ask for the secrecy of *e.g.* X'^y or Y'^x

Authentication

- "When Bob finishes the exchange, Alice and Bob agree on g^x and g^y"

Authentication

$$\underline{\mathrm{Exp}^{\mathrm{auth}}_{\mathcal{A}}(\eta)}$$

$G, g \leftarrow \mathrm{gen}_{\mathrm{DH}}(1^\eta)$

$pk_A, sk_A \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta)$

$pk_B, sk_B \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta)$

$x, y, X', Y', ok \leftarrow \perp$

$z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B)$

return $(ok = 1 \wedge (X' \neq g^x \ \vee \ Y' \neq g^y))$

- "When Bob finishes the exchange, Alice and Bob agree on g^x and g^y"

Authentication

$\mathsf{Exp}^{\mathsf{auth}}_{\mathcal{A}}(\eta)$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$
$pk_A, sk_A \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$
$pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$
$x, y, X', Y', ok \leftarrow \perp$
$z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B)$
return $(ok = 1 \wedge (X' \neq g^x \vee Y' \neq g^y))$

- "When Bob finishes the exchange, Alice and Bob agree on g^x and g^y"
- Advantage:
$$\mathsf{Adv}^{\mathsf{auth}}_{\mathcal{A}}(\eta) = \mathrm{P}\left[\mathsf{Exp}^{\mathsf{auth}}_{\mathcal{A}}(\eta) = 1\right]$$
- B authenticates A if for any PPTM $\mathcal{A}$,
$$\mathsf{Adv}^{\mathsf{auth}}_{\mathcal{A}}(\eta) \in \mathsf{negl}(\eta)$$

Anonymity

- “No one can learn who is running the protocol”

Anonymity

- "No one can learn who is running the protocol"
- An example of a privacy property
- Written as an indistinguishability game:
  distinguish whether A^0 or A^1 runs the protocol

Anonymity

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{anon},\beta}(\eta)}$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$

$pk_A^0, sk_A^0 \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$pk_A^1, sk_A^1 \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$

$x, y, X', Y', ok \leftarrow \perp$

$\beta' \leftarrow \mathcal{A}^{\mathcal{O}^\beta}(1^\eta, G, g, pk_A^0, pk_A^1, pk_B)$

return β'

$\mathcal{O}^\beta$ is $\{\mathcal{O}_{\mathsf{alice}}{}^\beta, \mathcal{O}_{\mathsf{bob}}\}$

- "No one can learn who is running the protocol"
- An example of a privacy property
- Written as an indistinguishability game: distinguish whether A^0 or A^1 runs the protocol

Anonymity

$$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{anon},\beta}(\eta)}$$

$G, g \leftarrow \mathsf{gen}_{\mathsf{DH}}(1^\eta)$
$pk_A^0, sk_A^0 \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$
$pk_A^1, sk_A^1 \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$
$pk_B, sk_B \leftarrow \mathsf{gen}_{\mathsf{sign}}(1^\eta)$
$x, y, X', Y', ok \leftarrow \perp$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}^\beta}(1^\eta, G, g, pk_A^0, pk_A^1, pk_B)$
return β'

$\mathcal{O}^\beta$ is $\{\mathcal{O}_{\mathsf{alice}}{}^\beta, \mathcal{O}_{\mathsf{bob}}\}$

- "No one can learn who is running the protocol"
- An example of a privacy property
- Written as an indistinguishability game: distinguish whether A^0 or A^1 runs the protocol
- Advantage:
  $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(\eta) = \left| \mathsf{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{anon},0}(\eta) = 1\right] - \mathsf{P}\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{anon},1}(\eta) = 1\right]\right|$
- Anonymity holds if for any PPTM $\mathcal{A}$,
  $$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(\eta) \in \mathsf{negl}(\eta)$$
- Though of course it does not hold here.

Proofs of security

Assumption

$$\begin{array}{l}
\mathrm{Exp}_{\mathcal{A}}^{\mathrm{CDH}}(\eta) \\
\hline
G, g \leftarrow \mathrm{gen}(1^\eta) \\
x \leftarrow\!\$ \; [\![0, |G| - 1]\!] \\
y \leftarrow\!\$ \; [\![0, |G| - 1]\!] \\
z \leftarrow \mathcal{A}(1^\eta, G, g, g^x, g^y) \\
\textbf{return } (z = g^{x \cdot y})
\end{array}$$

Goal

$$\begin{array}{l}
\mathrm{Exp}_{\mathcal{A}}^{\mathrm{secrecy}}(\eta) \\
\hline
G, g \leftarrow \mathrm{gen}_{\mathrm{DH}}(1^\eta) \\
pk_A, sk_A \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta, A) \\
pk_B, sk_B \leftarrow \mathrm{gen}_{\mathrm{sign}}(1^\eta, B) \\
z \leftarrow \mathcal{A}^{\mathcal{O}}(1^\eta, G, g, pk_A, pk_B) \\
\textbf{return } (z = g^{x \cdot y})
\end{array}$$

- Model the protocol, the properties, and the assumptions
- Proof by reduction
  e.g. "from $\mathcal{A}$ s.t. $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{secrecy}} \notin \mathrm{negl}$, we construct $\mathcal{B}$ s.t. $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{CDH}} \notin \mathrm{negl}$".
- Need to make sure $\mathcal{B}$ also runs in polynomial time

Questions?