INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

UMR IRISA

Introduction to Provable Security

Models, Adversaries, Reductions

Cryptography / Cryptology

- “from Greek κρυπτός *kryptós*, "hidden, secret"; and γράφειν *graphein*, "writing", or -λογία *-logia*, "study", respectively”
- “is the practice and study of techniques for secure communication in the presence of third parties (called adversaries).”

Source : www.wikipedia.org

SOME CRYPTOGRAPHIC GOALS

- Confidentiality
  - Content of conversation remains hidden
- Authenticity
  - Message is really sent by specific sender
- Integrity
  - Message has not been modified
- Privacy:
  - Sensitive (user) data remains hidden
- Covertcy
  - The fact that a conversation is taking place is hidden
- ….

Confidentiality

- Parties exchange messages
- Parties store documents (or strings e.g. passwords)

No unauthorized party can learn anything about contents.

Authenticity

- "Online": Alice proves legitimacy to Bob in real-time fashion (interactively)

No unauthorized party can impersonate a user

- "Offline": Alice generates proof of identity to be verified offline by Bob

No unauthorized party can forge the proof

Integrity

- Parties send or receive messages

No modification to content of message(s)

How cryptography works

- Use building blocks (primitives)
  - … either by themselves (hashing for integrity)
  - … or in larger constructions (protocols, schemes)

- Security must be guaranteed even if mechanism (primitive, protocol) is known to adversaries

- Steganography vs. cryptography:
  - Steganography: hide secret information in plain sight
  - Cryptography: change secret information to something else, then send it

A brief history

- "Stone age": secrecy of algorithm
  - Substitution and permutation (solvable by hand)
    - Caesar cipher, Vigenère cipher, etc.
- "Industrial Age": automation of cryptology
  - Cryptographic machines like Enigma
  - Fast, automated permutations (need machines to solve)
- "Contemporary Age": provable security
  - Starting from assumptions (e.g. a one-way function), I build a scheme, which is "provably" secure in model

Part II
The Provable Security Method

Security by trial-and-error

- Identify goal (e.g. confidentiality in P2P networks)
- Design solution – the strategy:
  - Propose protocol
  - Search for an attack
  - If attack found, fix (go to first step)
  - After many iterations or some time, halt
- Output: resulting scheme

- Problems:
  - What is “many” iterations/ “some” time?
  - Some schemes take time to break: MD5, RC4…

Provable security

- Identify goal. Define security:
  - Syntax of the primitive: e.g. algorithms (KGen, Sign, Vf)
  - Adversary (e.g. can get signatures for arbitrary msgs.)
  - Security conditions (e.g. adv. can't sign fresh message)
- Propose a scheme (instantiate syntax)
- Define/choose security assumptions
  - Properties of primitives / number theoretical problems
- Prove security – 2 step algorithm:
  - Assume we can break security of scheme (adv. $\mathcal{A}$)
  - Then build "Reduction" (adv. $\mathcal{B}$) breaking assumption

THE ESSENCE OF PROVABLE SECURITY

- Core question: what does “secure” mean?
  - “Secure encryption” vs. “Secure signature scheme”
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 1: Define your primitive (syntax)

Signature Scheme: algorithms (KGen, Sign, Vf)

* KGen(1^γ) outputs (sk, pk)

* Sign(sk,m) outputs S (prob.)

* Vf(pk,m,S) outputs 0 or 1 (det.)

THE ESSENCE OF PROVABLE SECURITY

- Core question: what does “secure” mean?
  - “Secure encryption” vs. “Secure signature scheme”
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 2: Define your adversary

Adversaries $\mathcal{A}$ can: know public information: γ, pk
get no message/signature pair
get list of message/signature pairs
submit arbitrary message to sign

THE ESSENCE OF PROVABLE SECURITY

- Core question: what does “secure” mean?
  - “Secure encryption” vs. “Secure signature scheme”
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 3: Define the security condition

Adversary $\mathcal{A}$ can output fresh (m,S) which verifies, with non-negligible probability (as a function of γ)

THE ESSENCE OF PROVABLE SECURITY

- Core question: what does "secure" mean?
  - "Secure encryption" vs. "Secure signature scheme"
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 4: Propose a protocol

Instantiate the syntax given in Step 1.

E.g. give specific algorithms for KGen, Sign, Vf.

The essence of provable security

- Core question: what does "secure" mean?
  - "Secure encryption" vs. "Secure signature scheme"
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 5: Choose security assumptions

For each primitive in the protocol, choose assumptions

- Security Assumptions (e.g. IND-CCA encryption)
- Number Theoretical Assumptions (e.g. DDH, RSA)

The essence of provable security

- Core question: what does "secure" mean?
  - "Secure encryption" vs. "Secure signature scheme"
- Say a scheme is secure against all known attacks
  - … will it be secure against a new, yet unknown attack?
- Step 6: Prove security

For each property you defined in steps 1-3:

- Assume there exists an adversary $\mathcal{A}$ breaking that security property with some probability ε
- Construct reduction $\mathcal{B}$ breaking some assumption with probability $f(\varepsilon)$

How reductions work

- Security assumptions are baseline

- Reasoning:
  - If our protocol/primitive is insecure, then the assumption is broken
  - But the assumption holds (by definition)
- Conclusion: The protocol cannot be insecure

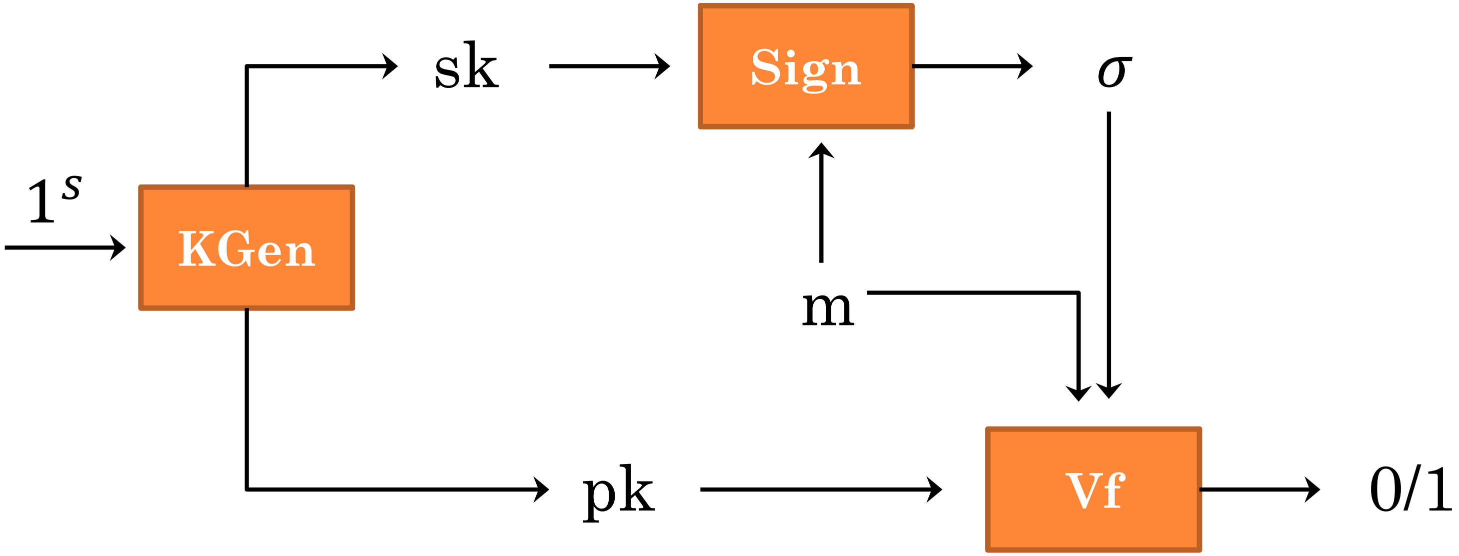
- Caveat:
  - Say an assumption is broken (e.g. DDH easy to solve)
  - What does that say about our protocol?

We don’t know!

Part III
Assumptions

We need computational assumptions

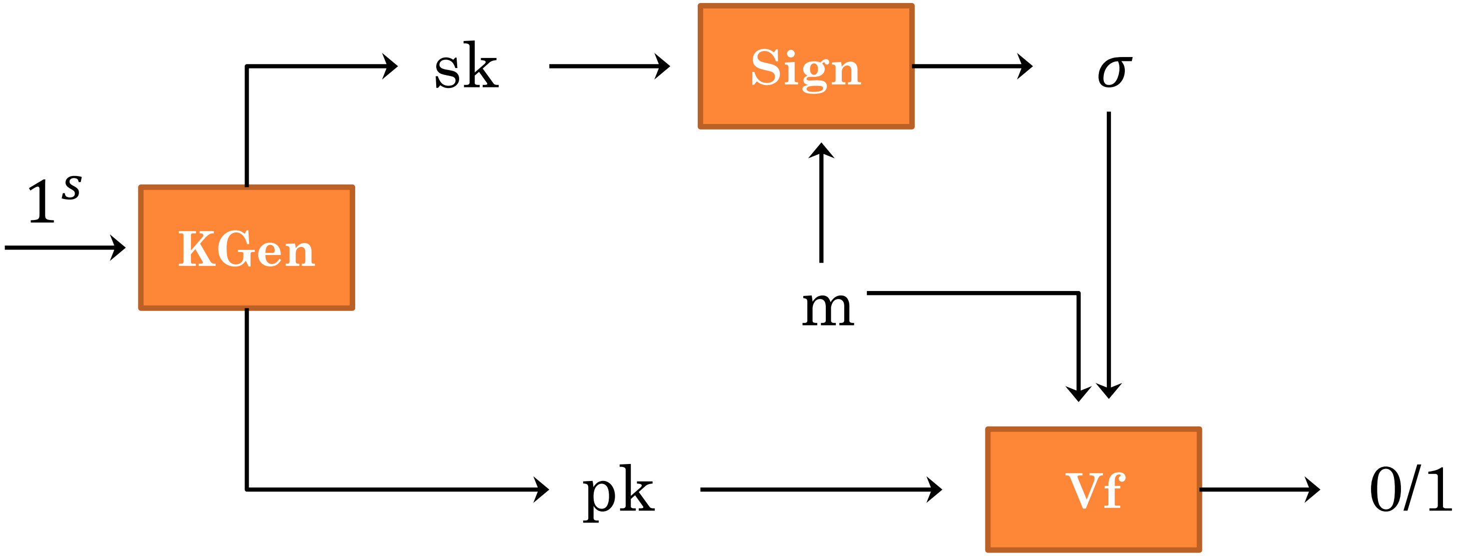
- Take our signature schemes (KGen, Sign, Vf)



- Correctness: if parameters are well generated, well-signed signatures always verify.

We need computational assumptions

- Take our signature schemes (KGen, Sign, Vf)

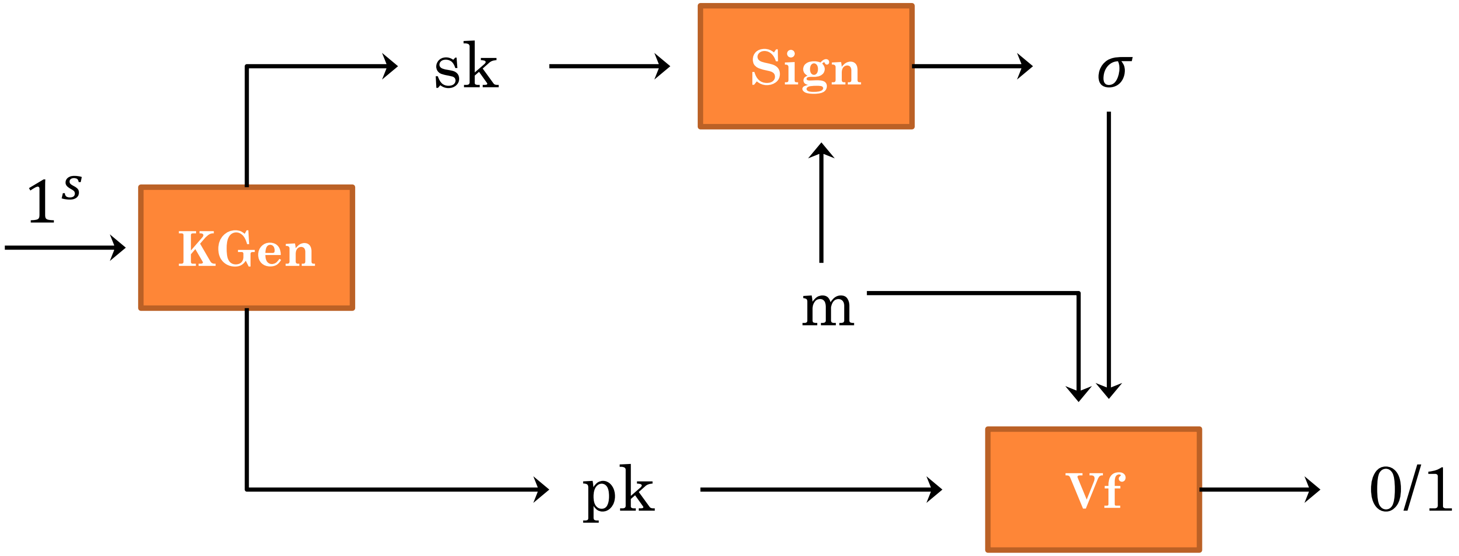


- Unforgeability: no adversary can produce signature for a fresh message m*

But any $\mathcal{A}$ can guess sk with probability ${}^{1}/_{2^{|sk|}}$

We need computational assumptions

- Take our signature schemes (KGen, Sign, Vf)



- Unforgeability: no adversary can produce signature for a fresh message m*

And any $\mathcal{A}$ can guess valid σ with probability $^1/_{2^{|\sigma|}}$

Some Computational Assumptions

- Of the type: It is “hard” to compute *x* starting from *y*.
- How hard?
  - Usually no proof that the assumption holds
  - Mostly measured with respect to “best attack”
  - Sometimes average-case, sometimes worst-case
- Relation to other assumptions:
  - A 1 “→” A 2: break A 2 => break A 1 **stronger**
  - A 1 “←” A 2: break A 1 => break A 2 **weaker**
  - A 1 “⇔” A 2: both conditions hold **equivalent**

EXAMPLES: DLOG, CDH, DDH

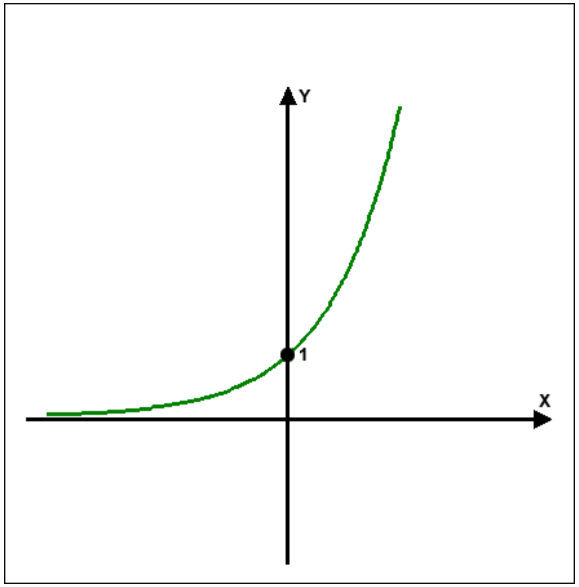
- Background:
  - Finite field $\mathbb{F}$, e.g. $\mathbb{Z}^*_p$ = {1, 2, … , p-1} for prime p
  - Multiplication, e.g. modulo p: $2(p-2) = 2p\ - 4 = p - 4$
  - Element g of prime order $q|\,(p-1)$:

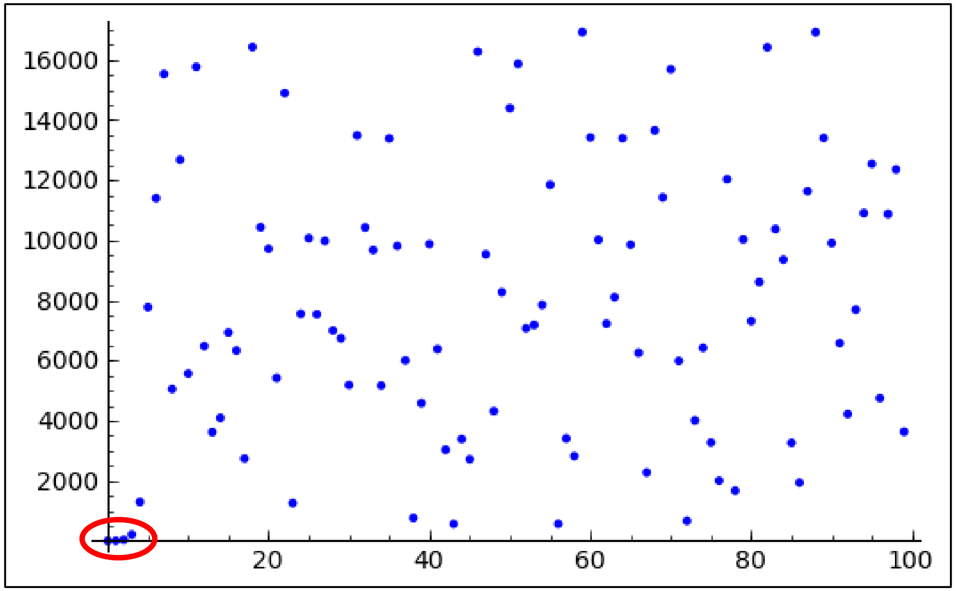
$$g^q = 1 \pmod p \ \ \text{AND} \ \ g^m \neq \ 1 \pmod p \ \ \forall\, m < q$$

  - Cyclic group $\mathrm{G} = < g > = \{1, g, g^2 \ldots g^{q-1}\}$

- DLog problem:
  - Pick $x\ \in_R\ \{1,\ \ldots\ ,\ q\}$. Compute $X =\ g^x \pmod p$.
  - Given (p, q, g, X) find x.
  - Assumed hard.

EXAMPLES: DLOG, CDH, DDH





- DLog problem:
  - Pick $x \in_R \{1, \ldots, q\}$. Compute $X = g^x \pmod p$.
  - Given (p, q, g, X) find x.
  - Assumed hard.

EXAMPLES: DLOG, CDH, DDH

- DLog problem:
  - Pick $x \in_R \{1, \ldots, q\}$. Compute $X = g^x \pmod{p}$.
  - Given (p, q, g, X) find x.
  - Assumed hard.
- CDH problem:
  - Pick $x, y \in_R \{1, \ldots, q\}$. Compute $X = g^x \pmod{p}$; $Y = g^y \pmod{p}$.
  - Given (p, q, g, X, Y) find g^{xy}.

Just to remind you: $g^{xy} = X^y = Y^x \neq XY = g^{x+y}$

- Solve D-LOG → Solve CDH
- Solve CDH $\nrightarrow$ Solve D-LOG

Examples: DLog, CDH, DDH

- DLog problem:
  - Pick $x \in_R \{1, \ldots, q\}$. Compute $X = g^x \pmod{p}$.
  - Given (p, q, g, X) find x.

- CDH problem:
  - Pick $x, y \in_R \{1, \ldots, q\}$. Compute $X = g^x \pmod{p}$; $Y = g^y \pmod{p}$.
  - Given (p, q, g, X, Y) find g^{xy}.

- DDH problem:
  - Pick $x, y, z \in_R \{1, \ldots, q\}$. Compute X, Y as above
  - Given (p, q, g, X, Y) distinguish g^{xy} from g^z.

How to solve the DLog problem

- In finite fields mod p:
  - Brute force (guess x) – $\mathcal{O}(q)$
  - Baby-step-giant-step: memory/computation tradeoff; $O(\sqrt{q})$
  - Pohlig-Hellman: small factors of q; $O(\log_p q\ (\log q + \sqrt{p}))$
  - Pollard-Rho (+PH): $O(\sqrt{p})$ for biggest factor p of q
  - NFS, Pollard Lambda, …
  - Index Calculus: $\exp((\ln q)^{\frac{1}{3}}(\ln(\ln(q)))^{\frac{2}{3}})$
- Elliptic curves
  - Generic: best case is BSGS/Pollard-Rho
  - Some progress on Index-Calculus attacks recently

Parameter Size vs. Security

ANSSI

Date	Sym.	RSA modulus	DLog Key	DLog Group	EC GF(p)	Hash
<2020	100	2048	200	2048	200	200
<2030	128	2048	200	2048	256	256
>2030	128	3072	200	3072	256	256

BSI

Date	Sym.	RSA modulus	DLog Key	DLog Group	EC GF(p)	Hash
2015	128	2048	224	2048	224	SHA-224+
2016	128	2048	256	2048	256	SHA-256+
<2021	128	3072	256	3072	256	SHA-256+

USING ASSUMPTIONS

- Implicitly used for all the primitives you have ever heard of
- Take ElGamal encryption:
  - Setup: N-bit prime q, L-bit prime p with $q \mid (p-1)$

    Generator g such that $\mathrm{Order}(g \bmod p) = q$

    $\boldsymbol{g^q = kp + 1}$ **for some** $\boldsymbol{k}$ **and** $\boldsymbol{g^m \neq np + 1}$ **for any** $\boldsymbol{n}$
  - Secret key: random $sk \in \{1, \dots, q-1\}$
  - Public key: $pk = g^{sk} \pmod{p}$

DLog: you can't compute *sk* from *pk*

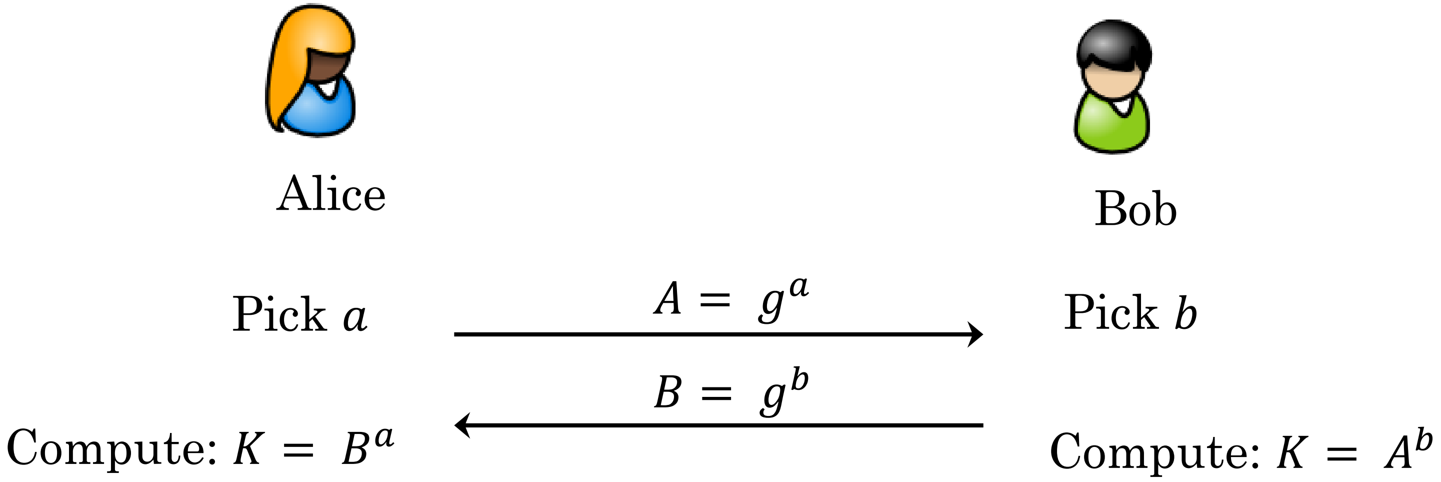
USING ASSUMPTIONS (2)

- Implicitly used for all the primitives you have ever heard of
- Take ElGamal encryption:
  - Setup: N-bit prime q, L-bit prime p with $q \mid (p-1)$
    Generator g such that $\mathtt{Order}(g \bmod p) = q$
  - Secret key: random $sk \in \{1, \ldots, q-1\}$
  - Public key: $pk = g^{sk} \pmod p$
  - Encryption: pick random r, output: $(g^r, M \cdot pk^r) \bmod p$
  - Decryption: $\frac{M \cdot pk^r}{(g^r)^{sk}} = \frac{M \cdot (g^{sk})^r}{(g^r)^{sk}}$

CDH: can't compute $g^{r \cdot sk}$ from g^r, g^{sk}

Using Assumptions (3)

- Implicitly used for all the primitives you have ever heard of
- Take Diffie-Helman key exchange (2-party):
  - Setup: p, q, g as before



Alice

Bob

Pick a

$A = g^a$

Pick b

$B = g^b$

Compute: $K = B^a$

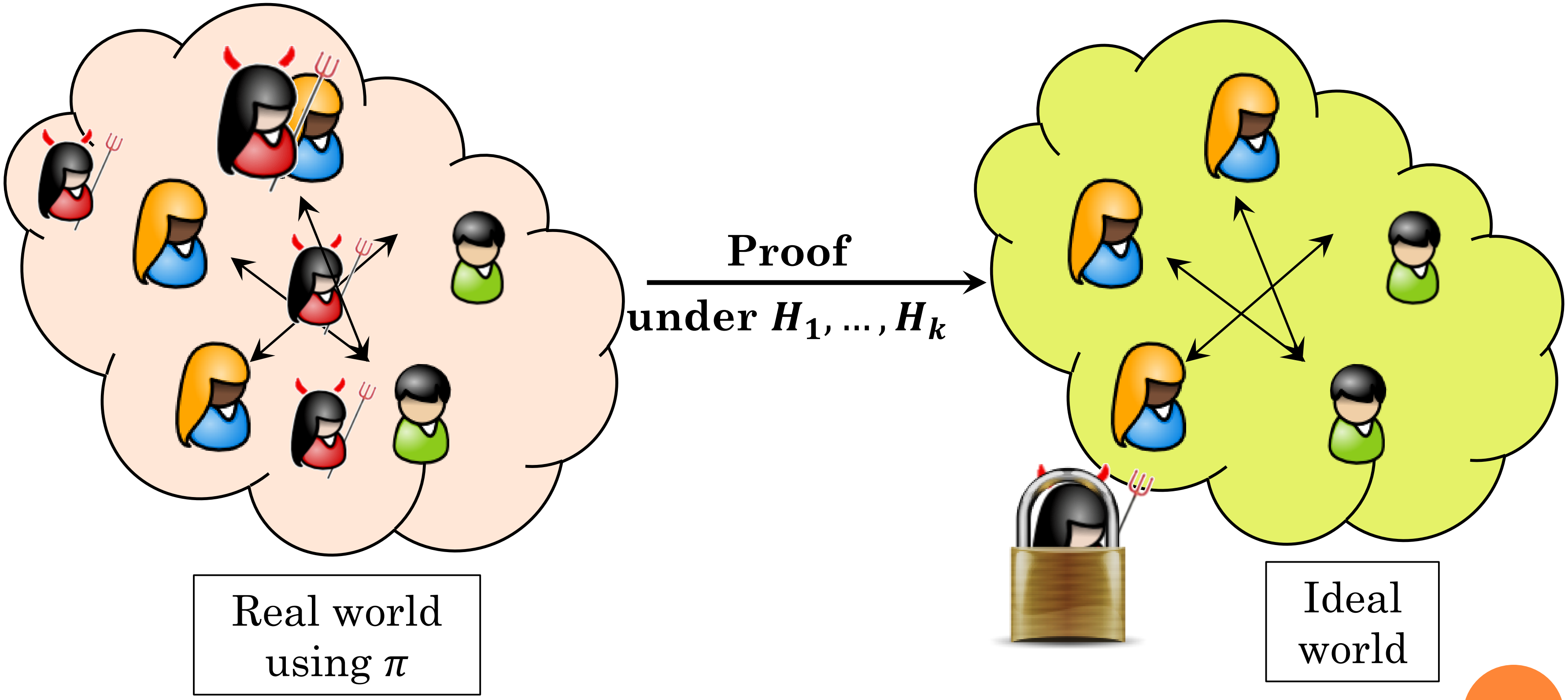
Compute: $K = A^b$

DDH: can't distinguish K from random, given A, B

Part IV
Security Models

Ideal Provable Security

- Given protocol π, assumptions $H_1, \ldots, H_k$



Proof

under $H_1, \ldots, H_k$

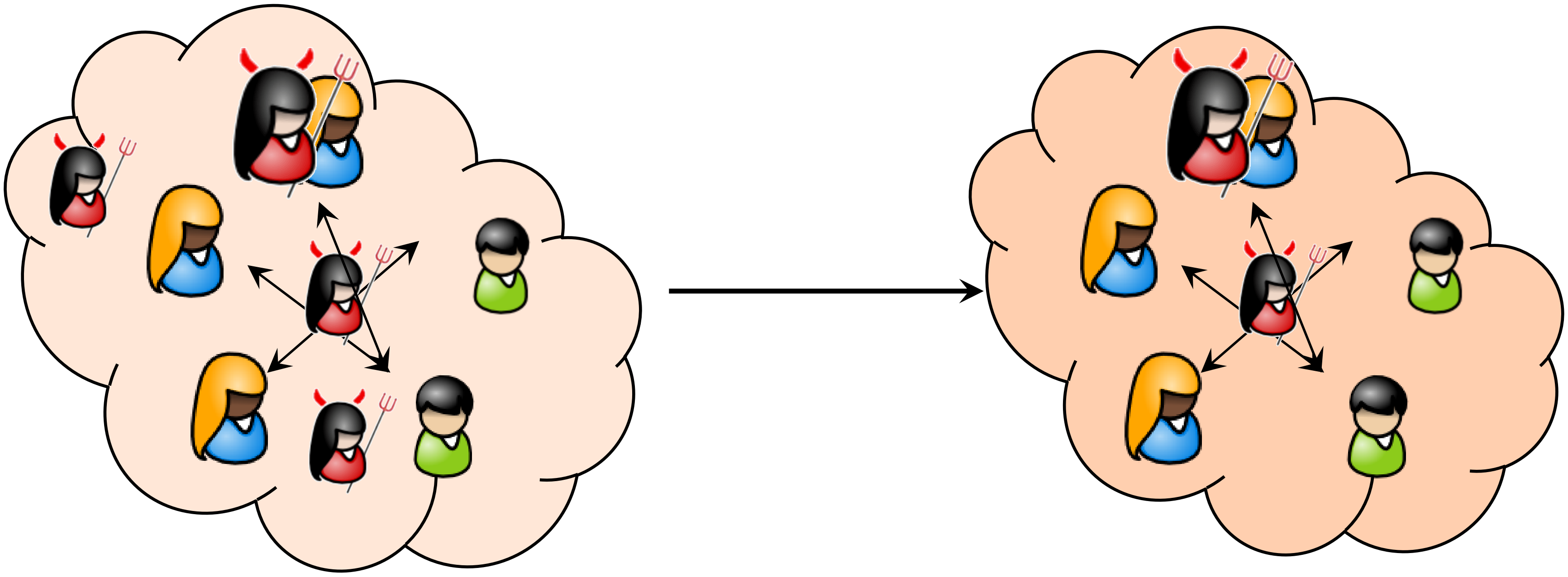
Real world using π

Ideal world

“Real World” is hard to describe mathematically

Provable Security

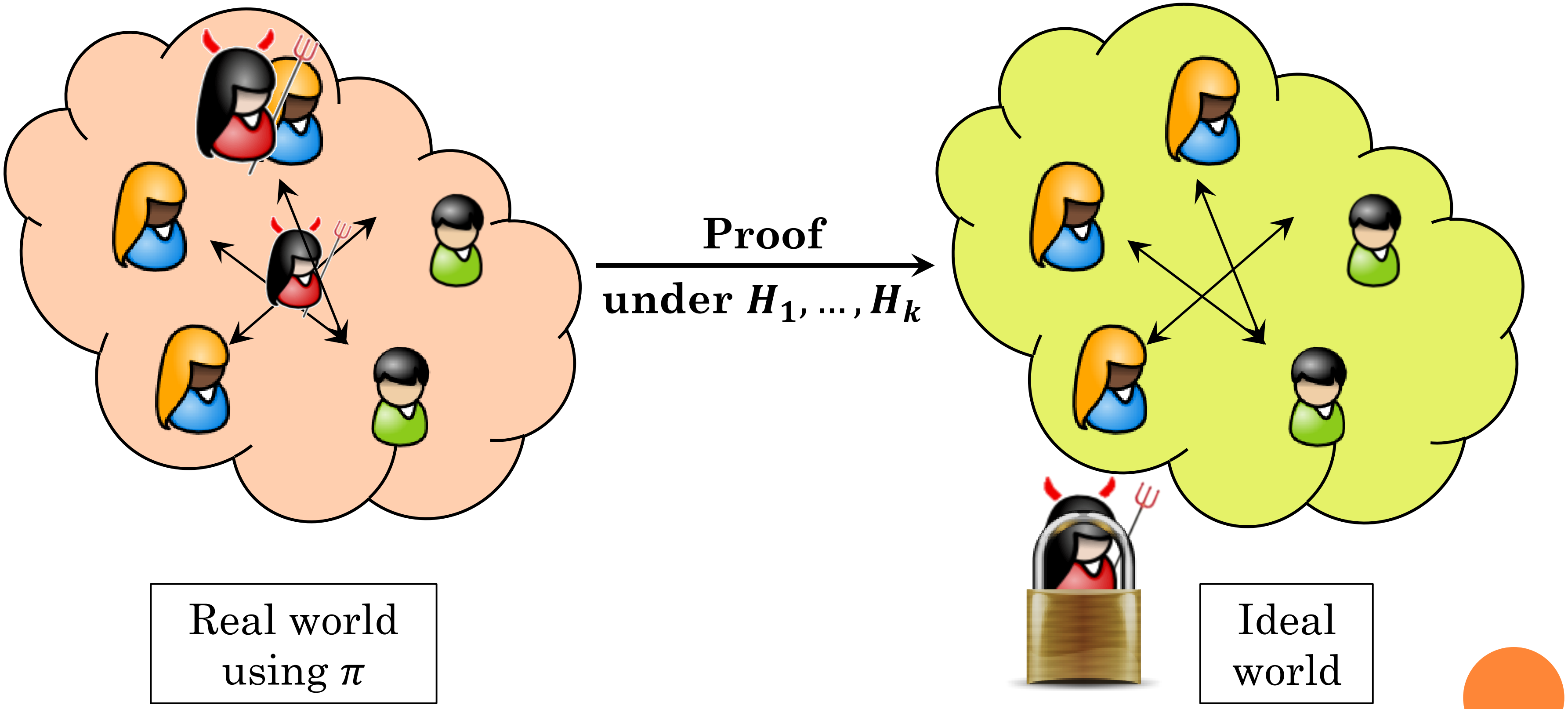
- Two-step process:



Real world using π

Modelled world using π

Provable Security



Proof

under $H_1, \ldots, H_k$

Real world using π

Ideal world

Components of Security Models

- Adversarial à-priori knowledge & computation:
  - Who is my adversary? (outsider, malicious party, etc.)
  - What does my adversary learn?
- Adversarial interactions (party-party, adversary-party, adversary-adversary – sometimes)
  - What can my adversary learn
  - How can my adversary attack?
- Adversarial goal (forge signature, find key, distinguish Alice from Bob)
  - What does my adversary want to achieve?

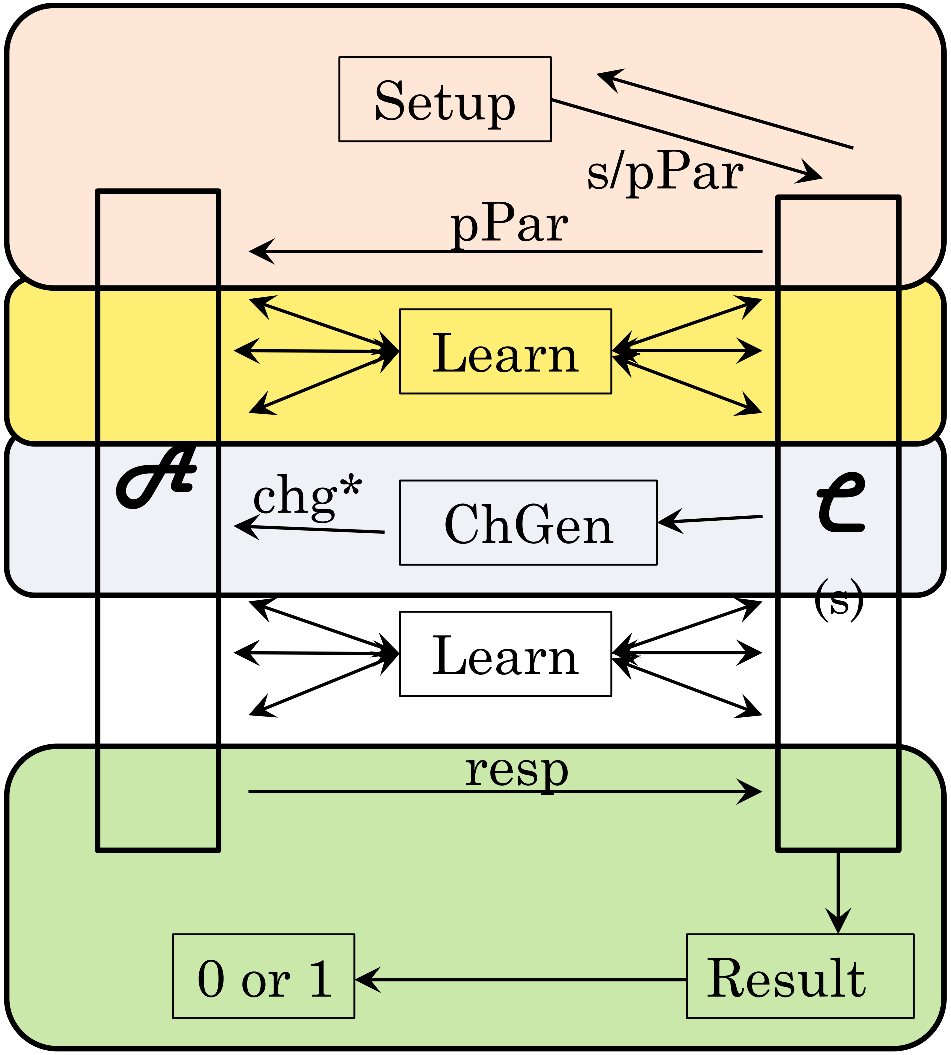
Game-Based Security

- Participants
  - Adversary $\mathcal{A}$ plays a game against a challenger $\mathcal{C}$
  - Adversary = attacker(s), has all public information
  - Challenger = all honest parties, has public information and secret information

- Attack
  - Oracles: $\mathcal{A}$ makes oracle queries to $\mathcal{C}$ to learn information
  - Test: special query by $\mathcal{A}$ to $\mathcal{C}$, to which $\mathcal{A}$ responds

    sometimes followed by more oracle queries
  - Win/Lose: a bit output by $\mathcal{C}$ at the end of the game

Canonical Game-Based Security



Game Structure

- Setup: generate game parameters s/pPar
- Learn: $\mathcal{A}$ queries oracles; $\mathcal{C}$ answers using s
- ChGen: $\mathcal{C}$ generates challenge chg*
- Result: $\mathcal{C}$ learns whether $\mathcal{A}$ has won or lost

EXAMPLE 1: SIGNATURE SCHEMES

- Intuition: a signature scheme $(\mathrm{KGen}, \mathrm{Sign}, \mathrm{Vf})$ is secure if and only if:
  - $\mathcal{A}$ should not be able to forge signatures

- Formal security definition: UNF-CMA

$$(sk, pk) \leftarrow \mathrm{KGen}(1^{\alpha})$$

$$(m, \sigma) \leftarrow A^{OSign(*)^{\mathrm{N}}}(1^{\alpha}, pk)$$

$$\text{Set : } \mathrm{L} = \{m_i, \sigma_i\}_{i=1,\ldots,\mathrm{N}} \text{ with } \sigma_i \leftarrow OSign(m_i)$$

$$A \text{ wins iff: } \mathrm{Vf}(pk, m, \sigma) = 1 \;\text{ and }\; \{m,*\} \notin \mathrm{L}$$

$OSign(*)$

On input m, set
$\sigma \leftarrow \mathrm{Sign}(sk, m)$

Output σ

Example 2: PKE

- Intuition: a PK encryption scheme $(\text{KGen}, \text{Enc}, \text{Dec})$ is secure if and only if:
  - $\mathcal{A}$ should not be able to learn encrypted messages

- Formally defining this (without decryptions):

$$\left[\begin{array}{l} (sk, pk) \leftarrow \text{KGen}(1^{\alpha}) \\ m \leftarrow_R M \\ c \leftarrow \text{Enc}(pk, m) \\ m' \leftarrow \text{A}(1^{\alpha}, pk, c) \\ \\ A \text{ wins iff: } m = m' \end{array}\right.$$

Example 2: PKE

- Intuition: a PK encryption scheme $(\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ is secure if and only if:
  - $\mathcal{A}$ should not be able to learn encrypted messages
- What if $\mathcal{A}$ can learn some ciphertext/plaintext tuples?

```
$(sk, pk) \leftarrow \mathrm{KGen}(1^{\alpha})$
$\mathrm{ready} \leftarrow A^{ODec(*)^{N}}(1^{\alpha}, pk)$
$m \leftarrow_{R} M$
$c \leftarrow \mathrm{Enc}(pk, m)$
$m' \leftarrow A^{ODec'(*)^{M}}(1^{\alpha}, pk, c)$

$A$ wins iff: $m = m'$
```

$ODec(*)$

On input c', output
$m \leftarrow \mathrm{Dec}(sk, c')$

$ODec'(*)$

On input $c' \neq c$, output
$m \leftarrow \mathrm{Dec}(sk, c')$
Else, output ב

Example 2: PKE

- Intuition: a PK encryption scheme (KGen, Enc, Dec) is secure if and only if:
  - $\mathcal{A}$ should not be able to learn encrypted messages

What if $\mathcal{A}$ can learn a single bit of the message?

1 bit can make a difference in a small message space!

  - $\mathcal{A}$ should not be able to learn even 1 bit of an encrypted message

Example 2: PKE

- Intuition: a PK encryption scheme $(\mathrm{KGen}, \mathrm{Enc}, \mathrm{Dec})$ is secure if and only if:
  - $\mathcal{A}$ must not learn even 1 bit of an encrypted message
- Formal definition: IND-CCA

$(sk, pk) \leftarrow \mathrm{KGen}(1^\alpha)$

$(m_1, m_2) \leftarrow A^{ODec(*)^N}(1^\alpha, pk)$

$b \leftarrow_R \{0,1\}$

$c \leftarrow \mathrm{Enc}(pk, m_b)$

$d \leftarrow A^{ODec'(*)^M}(1^\alpha, pk, c)$

A wins iff: $b = d$

$ODec(*)$

On input c', output
$m \leftarrow \mathrm{Dec}(sk, c')$

$ODec'(*)$

On input $c' \neq c$, output
$m \leftarrow \mathrm{Dec}(sk, c')$
Else, output ב

Measuring Adversarial Success

- Winning a game; winning condition:
  - Depends on relation R on $(*, <\text{game}>)$, with $<\text{game}> =$ full game input (of honest parties and $\mathcal{A}$)
  - Finally, $\mathcal{A}$ outputs x, wins if $(x, <\text{game}>) \in R$
- Success probability:
  - What is the probability that $\mathcal{A}$ "wins" the game?
  - What is the probability measured over? (e.g. randomness in $<\text{game}>$, sometimes probability space for keys, etc.)
- Advantage of Adversary:
  - How much better is $\mathcal{A}$ than a trivial adversary?

Trivial Adversaries

- Example 1: Signature unforgeability
  - $\mathcal{A}$ has to output a valid signature for message m
  - Trivial attacks: (1) guess signature (probability $2^{-|\sigma|}$)
    (2) guess secret key (probability $2^{-|sk|}$)
    (3) re-use already-seen σ
  - Goal: $\mathcal{A}$ outputs valid signature for **fresh** message m

- Example 2: Distinguish real from random
  - $\mathcal{A}$ has to output a single bit: real (0) or random (1)
  - Trivial attacks: (1) guess the bit (probability $^1/_2$)
    (2) guess secret key (probability $2^{-|sk|}$)

Adversarial Advantage

- Forgery type games:
  - $\mathcal{A}$ has to output a string of a “longer” size
  - Best trivial attacks: guess the string or guess the key
  - Advantage:

    $\mathrm{Adv}[A] = \mathrm{Prob}[A \text{ wins the game}]$

- Distinguishability-type games:
  - $\mathcal{A}$ must distinguish between 2 things: left/right, real/random
  - Best trivial attacks: guess the bit (probability $^1/_2$)
  - Advantage (different ways of writing it):

    $\mathrm{Adv}[A] = \mathrm{Prob}[A \text{ wins the game}] - {^1/_2}$

    $\mathrm{Adv}[A] = 2\,|\,\mathrm{Prob}[A \text{ wins the game}] - {^1/_2}\,|$

Defining Security

- Exact security definitions:
  - Input: number of significant queries of $\mathcal{A}$, execution time, advantage of $\mathcal{A}$
  - Example definition:

A signature scheme $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is (N, t, ε)-unforgeable under chosen message attacks (UNF-CMA) if for any adversary $\mathcal{A}$, running in time t, making at most N queries to the Signing oracle, it holds that:

$$\mathrm{Adv}[A] := \mathrm{Prob}[A \text{ wins the game}] \leq \ \varepsilon$$

  - If a scheme is $(N, t, 1)$-UNF-CMA, then the scheme is insecure!

Defining Security

- Asymptotic security:
  - Consider behaviour of ε as a function of the size of the security parameter 1^α:

A signature scheme $(\mathrm{KGen}, \mathrm{Sign}, \mathrm{Vf})$ is (N, t, ε)-unforgeable under chosen message attacks (UNF-CMA) if for any adversary $\mathcal{A}$, running in time t, making at most N queries to the Signing oracle, it holds that:

$$\mathrm{Adv}[A] := \mathrm{Prob}[A \text{ wins the game}] \leq \varepsilon$$

The signature is (N, t)-unforgeable under chosen message attacks if for any adversary $\mathcal{A}$ as above, it holds:

$$\mathrm{Adv}[A] \leq \mathrm{negl}(1^\alpha)$$

Simulation-Based Definitions

- Game-based definitions
  - Well understood and studied
  - Can capture attacks up to "one bit of information"
  - What else do we need?

- Zero-Knowledge: "nothing leaks about…"
  - Real world: "real" parties, running protocol in the pre-sence of a "local" adversary
  - Ideal world: "dummy" parties, simulator that formalizes the most leakage allowed from the protocol
  - "Global" adversary: distinguisher real/ideal world – if simulator is successful, then real world leaks as much as ideal world

Security Models – Conclusions

- Requirements:
  - Realistic models: capture "reality" well, making proofs meaningful
  - Precise definitions: allow quantification/classification of attacks, performance comparisons for schemes, generic protocol-construction statements
  - Exact models: require subtlety and finesse in definitions, in order to formalize slight relaxations of standard definitions

- Provable security is an art, balancing strong security requirements and security from minimal assumptions

Part V
Proofs of Security

Game Hopping

- Start from a given security game G_0
- Modify G_0 a bit (limiting $\mathcal{A}$) to get G_1
- Show that for protocol π, games G_0 and G_1 are equivalent (under assumption A), up to negligible factor ε_1:

$$\boxed{\boldsymbol{G_0 \cong_{\varepsilon_1} G_1}: \quad \text{Prob}[A \text{ wins } G_0] \leq \text{Prob}[A \text{ wins } G_1] + \varepsilon_1}$$

- Hop through $G_2, G_3, \ldots, G_n$ (such that $G_{i-1} \cong_{\varepsilon_i} G_i$ for all i)
- For last game G_n find Prob[A wins G_n]; then:

$$\boxed{\text{Prob}[A \text{ wins } G_0] \leq \sum_{i=1}^{n} \varepsilon_i + \text{Prob}[A \text{ wins } G_n]}$$

PROVING $G_{i-1} \cong_{\varepsilon_i} G_i$

- Method 1: Reduce game indistinguishability to assumption or hard problem
  - If there exists a distinguisher $\mathcal{A}$ between G_{i-1} and G_i winning with probability ${}^{1}/_{2} + \ \delta$ then there exists an adversary $\mathcal{B}$ against assumption H_1 winning with probability $\delta' = f(\delta)$
  - So, Prob[A wins G_0] – Prob[A wins G_1] $\leq \ \delta + \ \delta' =: \ \varepsilon_1$
- Method 2: Reduce "difference" between games to assumption or hard problem
  - By construction, $\mathcal{A}$ can win G_0 more easily than G_1 (since $\mathcal{A}$ is more limited in G_1)
  - If there exists an adversary B that can "take advantage of" the extra ability it has in G_0 to win w.p. Prob[A wins G_1] $+ \delta$, then there exists B against H_1 winning w.p. δ'… (as above)

Game Equivalence & Reductions

- Reduction: algorithm $\mathcal{R}$ taking adversary $\mathcal{A}$ against a game, outputting adversary $\mathcal{B}$ against another game/hard problem

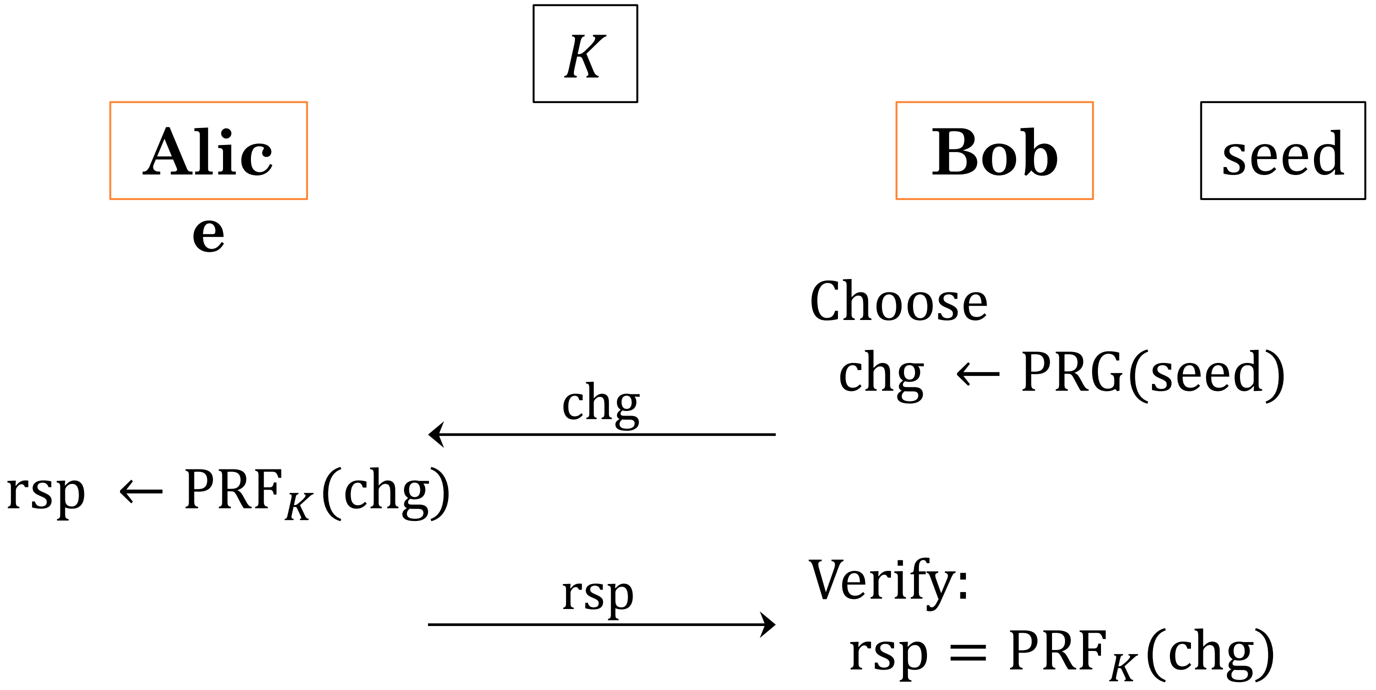
$$\mathcal{R}^{\mathcal{A}} \rightarrow \mathcal{B}$$

- Intuition: if there exists an adversary $\mathcal{A}$ against game G, this same adversary can be used by $\mathcal{R}$ to obtain $\mathcal{B}$ against G'
- $\mathcal{A}$ interacts with challenger $\mathcal{C}$ in G, $\mathcal{B}$ interacts with $\mathcal{C}'$ in G'
- In order to fully use $\mathcal{A}$, $\mathcal{B}$ needs to simulate $\mathcal{C}$:
  - $\mathcal{A}$ queries $\mathcal{C}$ in game G: $\mathcal{B}$ must answer query
  - $\mathcal{A}$ sends challenge input to $\mathcal{C}$: $\mathcal{B}$ must send challenge
  - $\mathcal{A}$ answers challenge: $\mathcal{B}$ uses response in game G'

Part VI
An Example

Secure Symmetric-key Authentication

- Alice wants to authenticate to Bob, with whom she shares a secret key



K

Alic
e

Bob

seed

Choose
$\text{chg} \leftarrow \text{PRG}(\text{seed})$

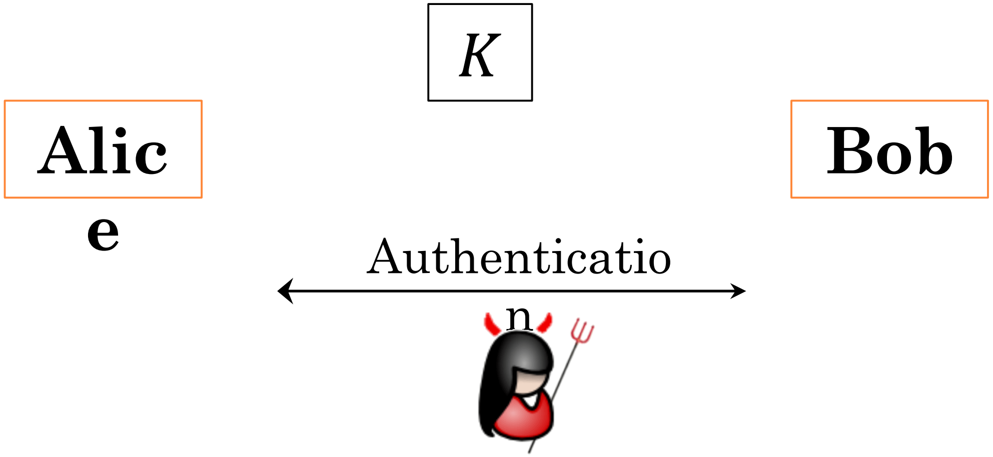
chg

$\text{rsp} \leftarrow \text{PRF}_K(\text{chg})$

rsp

Verify:
$\text{rsp} = \text{PRF}_K(\text{chg})$

Security of Authentication



K

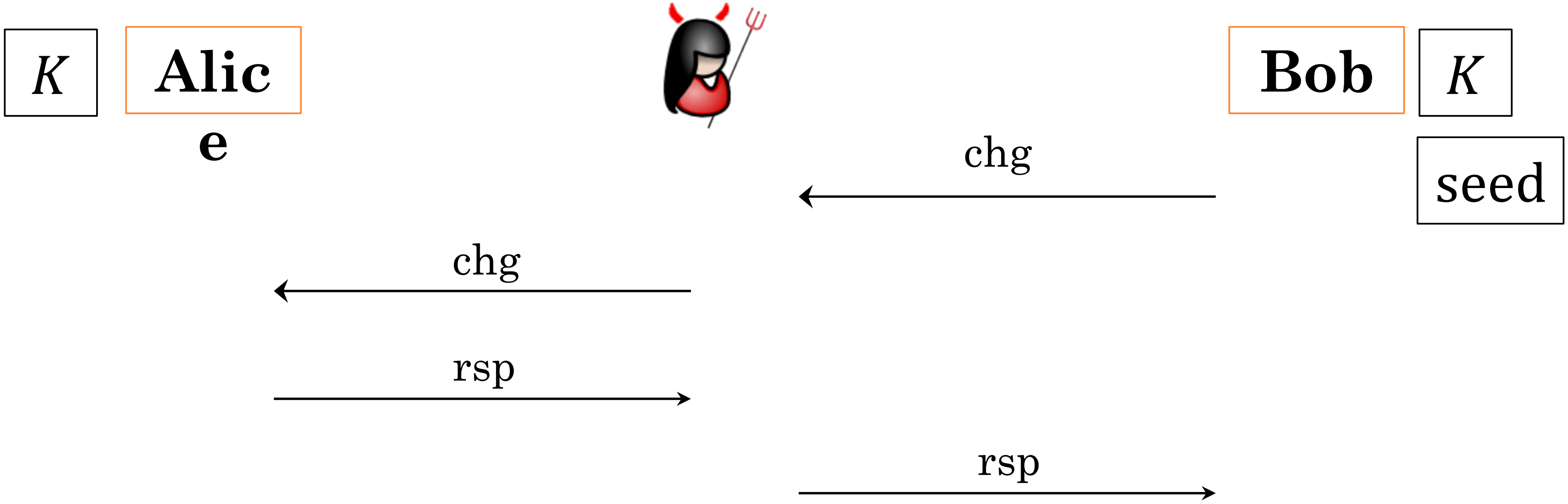
Alice

Bob

Authentication

- Nobody but Alice must authenticate to Bob
  - Who is my adversary?
    - A man-in-the-middle
  - What can they do?
    - Intercept messages, send messages (to Alice or Bob), eavesdrop
  - What is they goal of $\mathcal{A}$?
    - Make Bob accept $\mathcal{A}$ as being Alice

Trivial Attacks: Relay



- Relay attacks bypass any kind of cryptography: encryption, hashing, signatures, etc.
- Countermeasure: distance bounding (we’ll see it later)

Secure Authentication: Definition

- Session ID: tuple $< \mathrm{chg}, \mathrm{rsp} >$ used between partners
- Oracles:
  - NewSession($*$): input either P_1 = Alice or P_2 = Bob
    outputs session "handle" π
  - Send($*$,$*$): input handle π and message $m \in M \cup \{\mathrm{Prompt}\}$
    transmits m to partner in π, outputs m'
  - Result($*$): input a handle π with partner P_2
    outputs 1 if P_2 accepted authentication in π,
    0 if P_2 rejected, and $\beth$ otherwise

Secure Authentication: Game

Game ImpSec:

$k \leftarrow_R \text{KSpace}(1^\alpha)$
seed $\leftarrow_R$ SSpace (1^α)
done $\leftarrow A^{\text{NewSession}(*),\text{Send}(*,*),\text{Result}(*)}(1^\propto)$

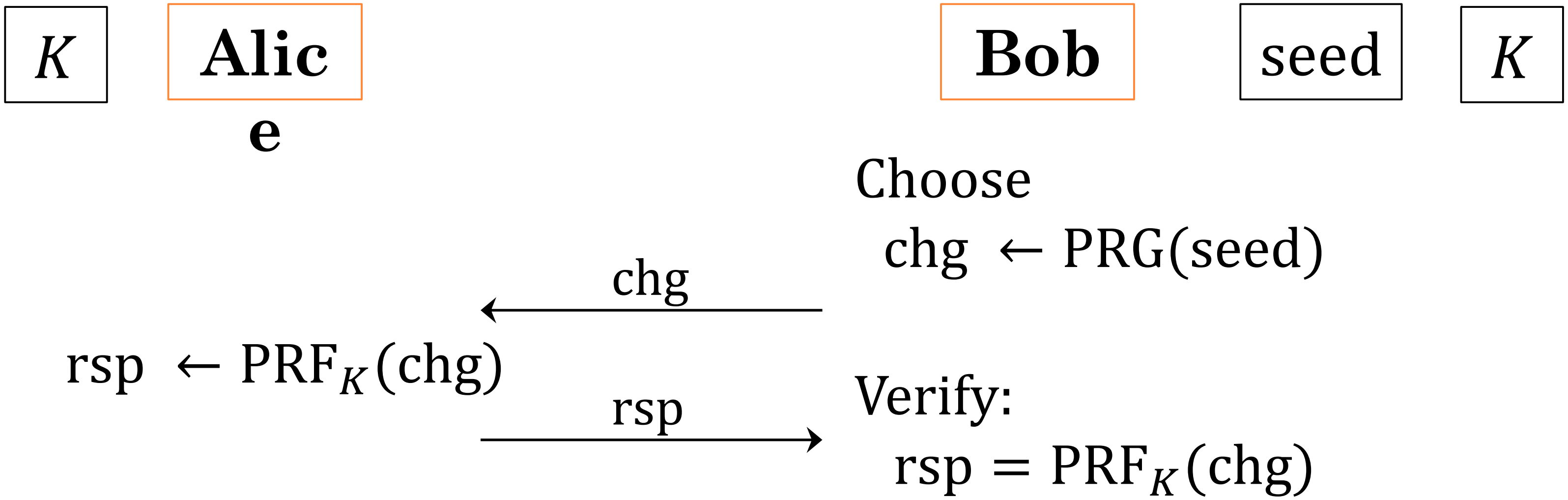
$\mathcal{A}$ wins iff $\exists\, \pi$ output by NewSession(P_2) such that:

- Result$(\pi) = 1$;
- There exists no π' output by NewSession(P_1) such that sid(π) = sid(π')

- Protocol is (N_1, N_2, ε)-impersonation secure iff. no adversary $\mathcal{A}$ using N_i sessions with P_i wins w.p. $\geq \varepsilon$.

$$\text{Adv}[A] \coloneqq \text{Prob}[A \text{ wins}]$$

PRGs and PRFs



$\boxed{K}$ **Alic**
e

Bob seed $\boxed{K}$

Choose
chg $\leftarrow$ PRG(seed)

chg

rsp $\leftarrow \mathrm{PRF}_K(\mathrm{chg})$

rsp

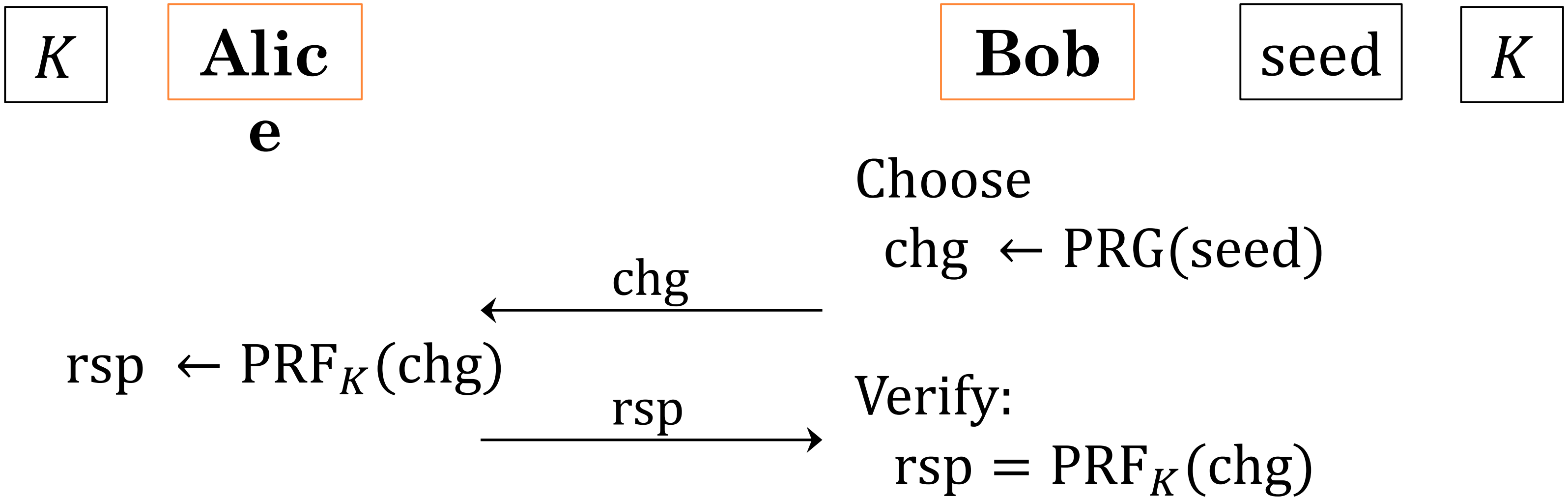
Verify:
rsp $= \mathrm{PRF}_K(\mathrm{chg})$

- Pseudorandomness of PRG:

key $\leftarrow_R$ Kspace
$d \leftarrow A^{\mathrm{Eval}_b()}$
A wins iff. $d = b$

```
Eval_b():
    if $b = 0$, return Rand()
    else, return PRG(key)
```

PRGs and PRFs



K

Alic
e

Bob

seed

K

Choose
$\mathrm{chg} \leftarrow \mathrm{PRG}(\mathrm{seed})$

chg

$\mathrm{rsp} \leftarrow \mathrm{PRF}_K(\mathrm{chg})$

rsp

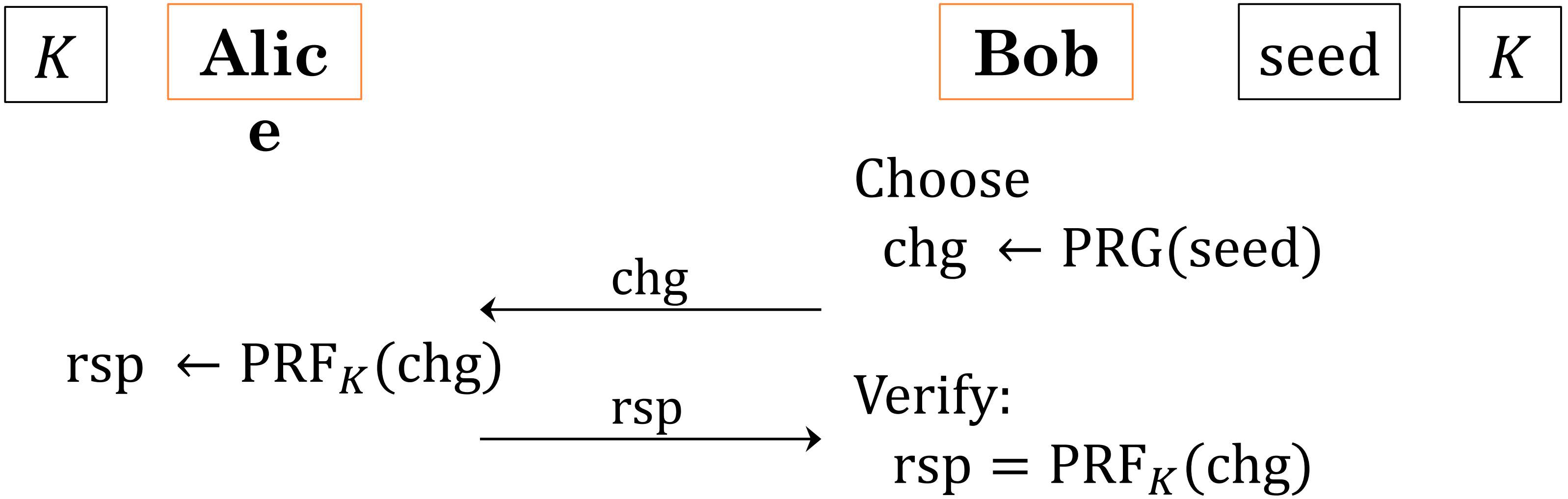
Verify:
$\mathrm{rsp} = \mathrm{PRF}_K(\mathrm{chg})$

- Pseudorandomness of PRF:

```
    $\mathrm{key} \leftarrow_R \mathrm{Kspace}$
    $d \leftarrow A^{\mathrm{Eval}_b()}$
$A$ wins iff. $d = b$
```

```
$\mathrm{Eval}_b()$:
    choose $x \leftarrow_R \mathrm{X}$
    if $b = 0$, return Rand(x)
    else, return $\mathrm{PRF}_{\mathrm{key}}(x)$
```

Proving Security



K

Alic
e

Bob

seed

K

Choose
chg ← PRG(seed)

chg

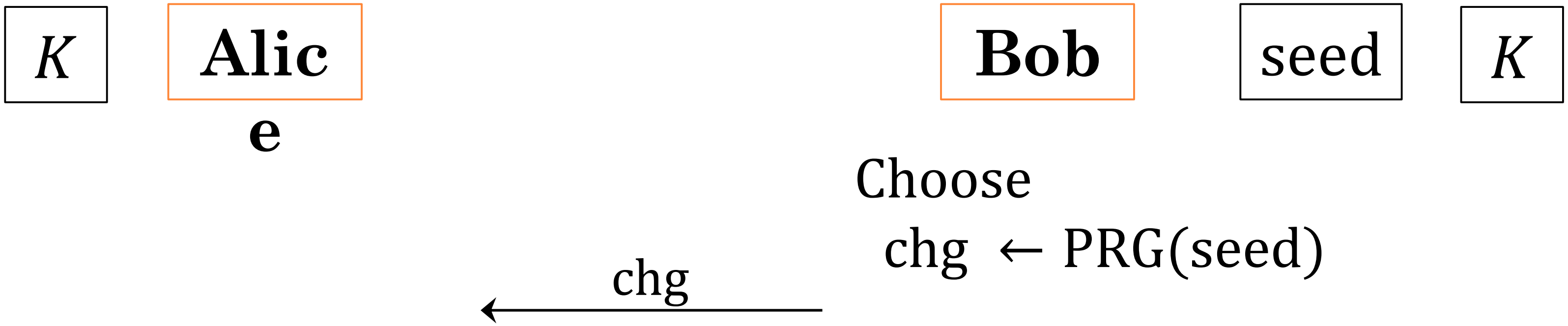
rsp ← $\mathrm{PRF}_K(\mathrm{chg})$

rsp

Verify:
rsp = $\mathrm{PRF}_K(\mathrm{chg})$

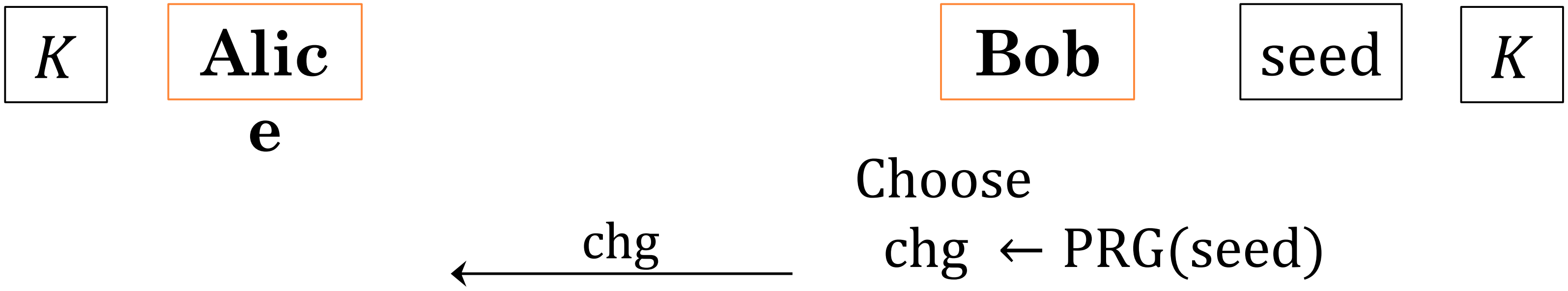
- Intuition:
  - If the PRG is good, then each chg is (almost) unique (up to collisions)
  - If the PRF is good, then each rsp looks random to adversary
  - Unless adversary relays, no chance to get right answer

PROVING SECURITY



- Proof, step 1:
  - Game G_0: Game ImpSec
  - Game G_1: Replace chg output by P_2 by random
  - Equivalence: $G_0 \cong G_1$: if there exists ε-distinguisher $\mathcal{A}$ between G_0 and G_1, then there exists $\mathcal{B}$ against PRG winning w.p. ε
    - Basically the intuition is that if $\mathcal{A}$ can distinguish between the two games, he can distinguish real (PRG) from truly random challenges

Proving Security



K

Alic
e

Bob

seed

K

Choose

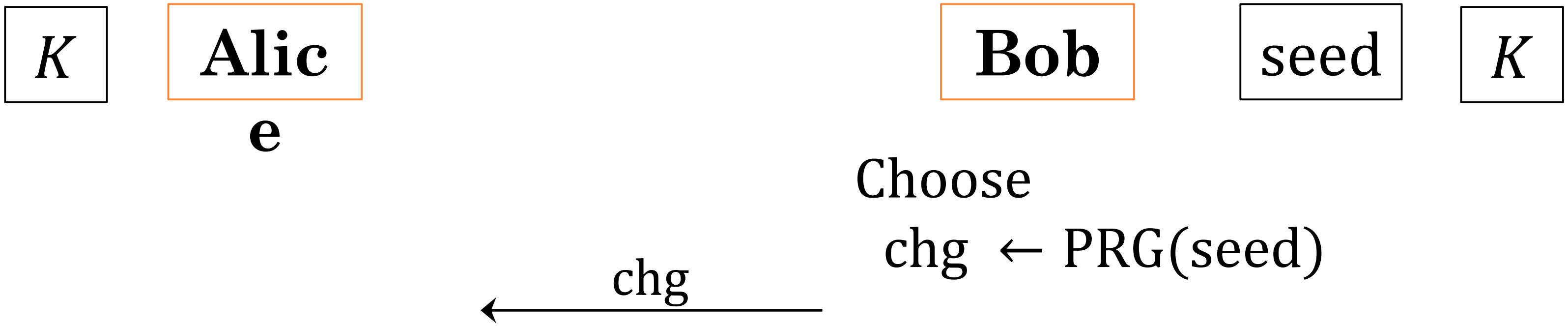
chg $\leftarrow$ PRG(seed)

chg

- Proof, equivalence $G_0 \cong G_1$:
  - $\exists$ ε-distinguisher $\mathcal{A}$ for G_0 / G_1 $\Rightarrow$ $\exists$ $\mathcal{B}$ winning PRG w.p. ε
    - Simulation: $\mathcal{B}$ chooses key $K \leftarrow_R$ KSpace and simulate any requests to Send(π, Prompt) by Eval$_b$() queries in PRG game
    - Finally $\mathcal{A}$ guesses either game G_0 ($\mathcal{B}$ outputs 1) or G_1 ($\mathcal{B}$ outputs 0)

```
Game PRG                          | Eval_b():
    $seed \leftarrow_R$ Kspace    |     if $b = 0$, return Rand()
    $d \leftarrow B^{\text{Eval}_b()}$ |     else, return PRG(key)
$B$ wins iff. $d = b$
```

Proving Security



K

Alice

Bob

seed

K

Choose

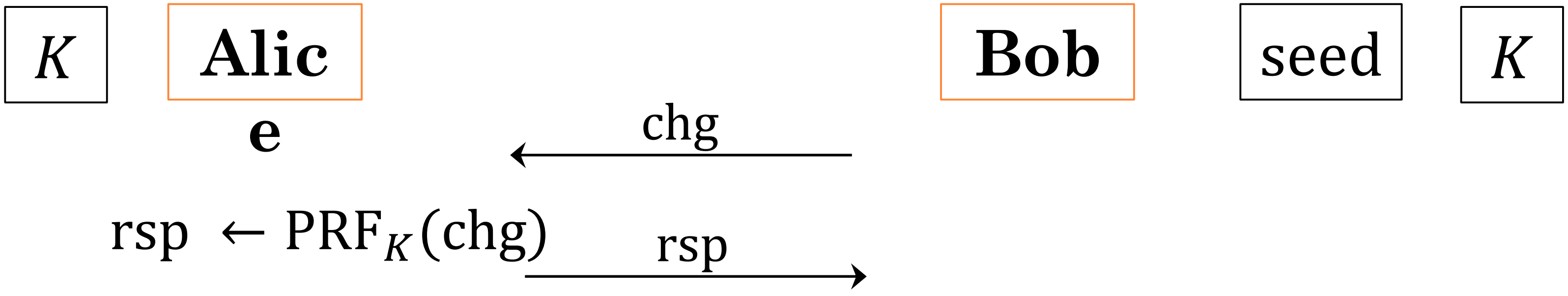
chg ← PRG(seed)

chg

- Proof, step 2:
  - Game G_0: Game ImpSec
  - Game G_1: Replace chg output by P_2 by random
  - Game G_2: Abort if collision in chg
  - Equivalence: $G_1 \cong G_2$: collisions in random strings occur in 2 different sessions w.p. $({}^{1}/{}_{2})^{|\text{chg}|}$. But we have a total of N_2 sessions, so the total probability of a collision is:

$$\binom{N_2}{2}\ 2^{-|\text{chg}|}$$

PROVING SECURITY



K

Alic
e

Bob

seed

K

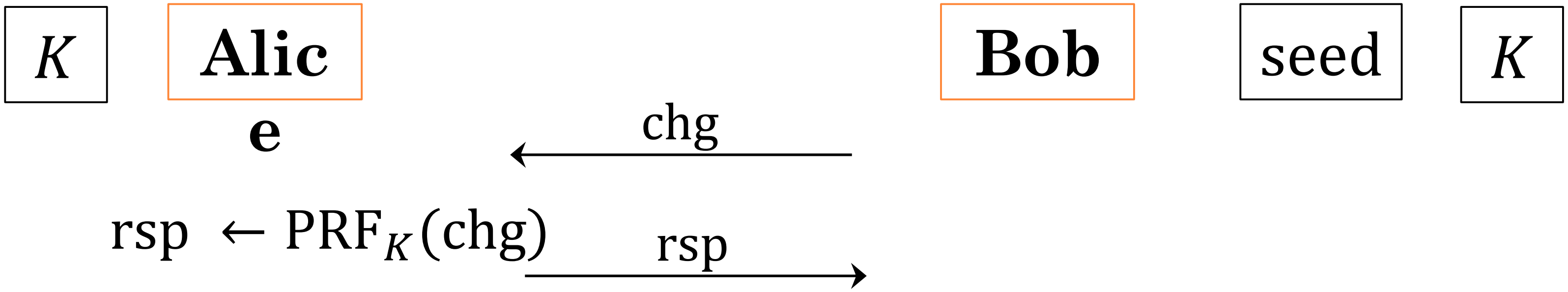
chg

$\mathrm{rsp} \leftarrow \mathrm{PRF}_K(\mathrm{chg})$

rsp

- Proof, step 3:
  - Game G_0: Game ImpSec
  - Game G_1: Replace chg output by P_2 by random
  - Game G_2: Abort if collision in chg
  - Game G_3: replace honest responses by consistent, truly random strings
  - Equivalence: $G_2 \cong G_3$: Similar to reduction to PRG, only this time it is to the pseudorandomness of the PRF.

Proving Security



K

Alic
e

Bob

seed

K

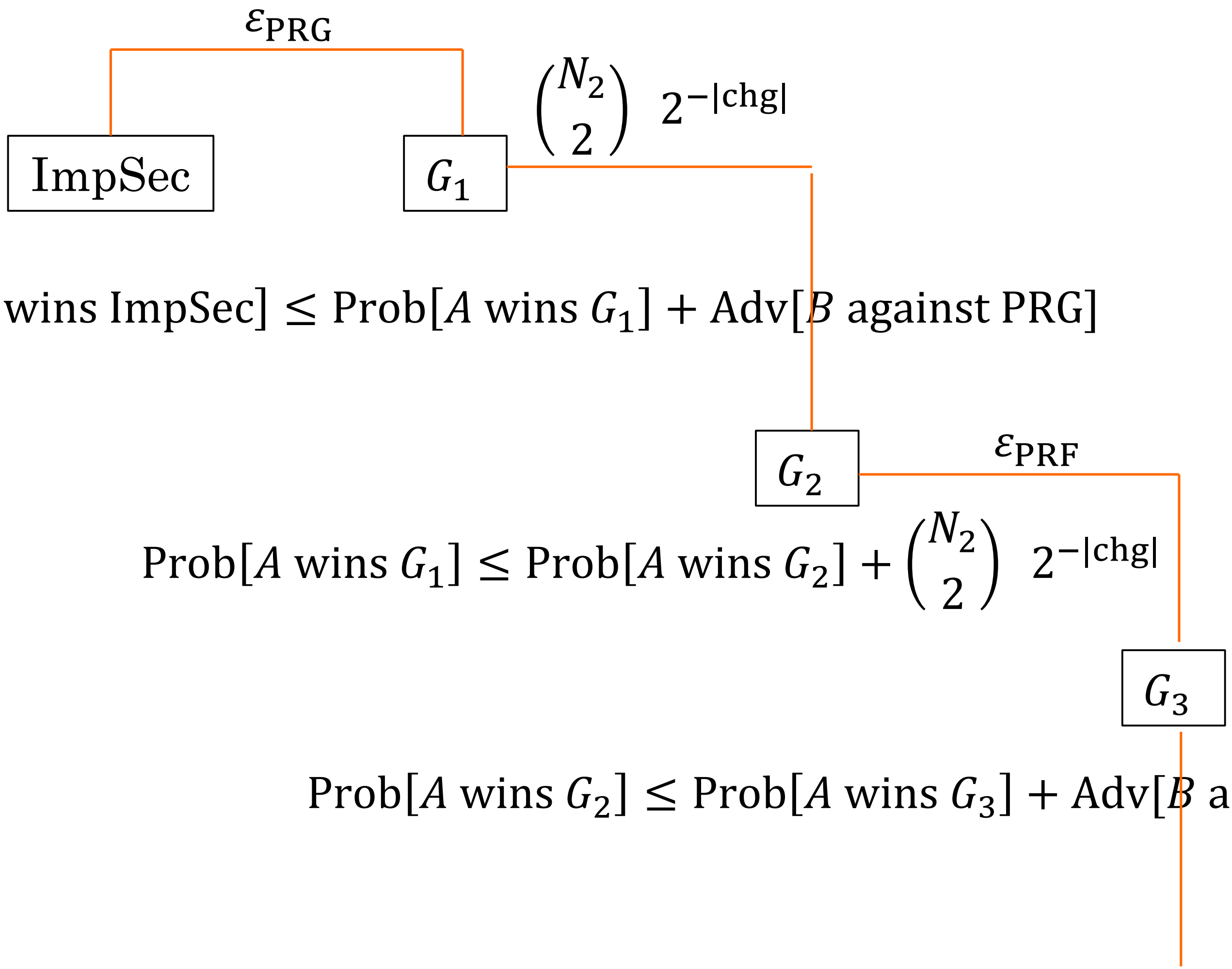
chg

$\mathrm{rsp} \leftarrow \mathrm{PRF}_K(\mathrm{chg})$

rsp

- Proof, step 4:
  - Game G_0: Game ImpSec
  - Game G_1: Replace chg output by P_2 by random
  - Game G_2: Abort if collision in chg
  - Game G_3: replace honest responses by consistent, truly random strings
  - At this point, the best the adversary can do is to guess a correct chg/rsp, i.e. $\mathrm{Prob}[A \text{ wins } G_3] = N_1 \cdot 2^{-|\mathrm{chg}|} + N_2 \cdot 2^{-|\mathrm{rsp}|}$

Putting It Together



ε_{PRG}

ImpSec

G_1

$\binom{N_2}{2}\ 2^{-|\text{chg}|}$

$$\text{Prob}[A \text{ wins ImpSec}] \le \text{Prob}[A \text{ wins } G_1] + \text{Adv}[B \text{ against PRG}]$$

G_2

ε_{PRF}

$$\text{Prob}[A \text{ wins } G_1] \le \text{Prob}[A \text{ wins } G_2] + \binom{N_2}{2}\ 2^{-|\text{chg}|}$$

G_3

$$\text{Prob}[A \text{ wins } G_2] \le \text{Prob}[A \text{ wins } G_3] + \text{Adv}[B \text{ against PRF}]$$

$$\text{Prob}[A \text{ wins } G_3] = N_1 \cdot 2^{-|\text{chg}|} + N_2 \cdot 2^{-|\text{rsp}|}$$

Security Statement

- For every (N_1, N_2, ε)- impersonation security adver-sary $\mathcal{A}$ against the protocol, there exist:
  - An ε_{PRG}-distinguisher against PRG
  - An ε_{PRF}-distinguisher against PRF

  such that:

$$\varepsilon \leq \varepsilon_{\text{PRG}} + \varepsilon_{\text{PRF}} + \binom{N_2}{2}\ 2^{-|chg|} + N_1 \cdot 2^{-|chg|}\ +\ N_2 \cdot 2^{-|rsp|}$$

Part VII Conclusions

Provable Security

- Powerful tool
- We can prove that a protocol is secure by design
- Captures generic attacks within a security model
- Can compare different schemes of same “type”
- 3 types of schemes:
  - Provably Secure
  - Attackable (found an attack)
  - We don’t know (unprovable, but not attackable)