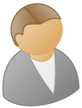
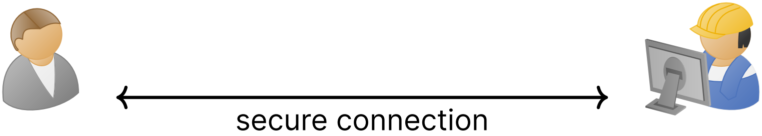
How do I authenticate myself in a post-quantum world?

Peeter Laud

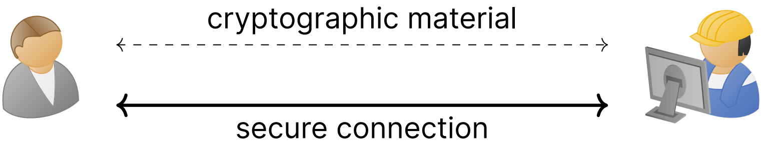
Why and how do I authenticate myself?



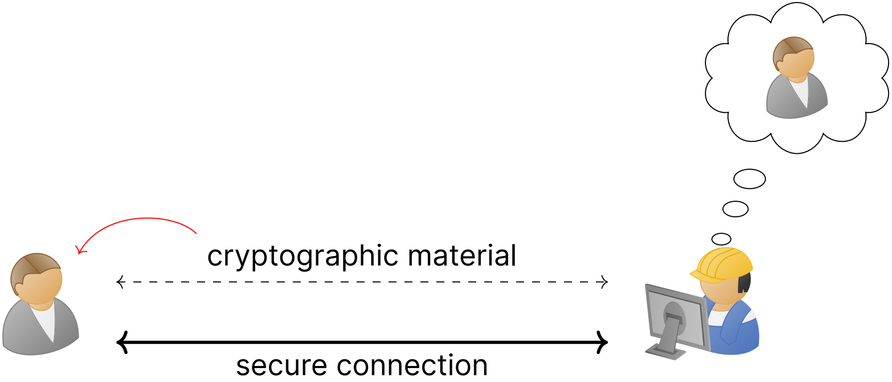
Why and how do I authenticate myself?



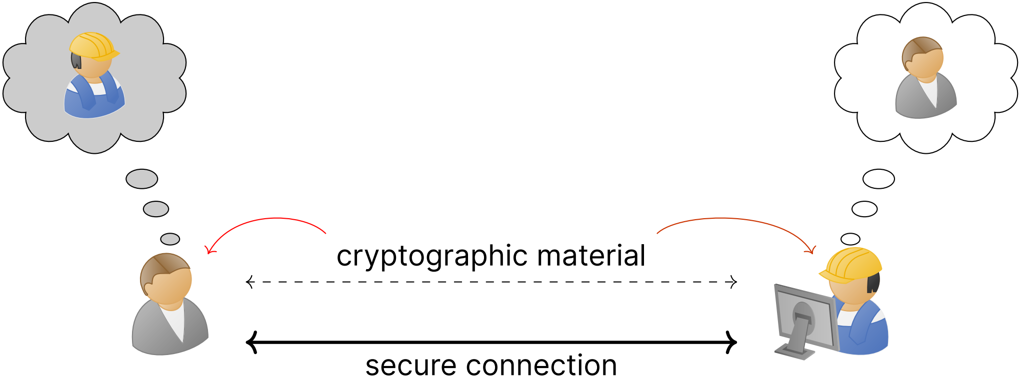
Why and how do I authenticate myself?



Why and how do I authenticate myself?



Why and how do I authenticate myself?



Identifying myself

What is "me"?

- Name?
- Date of birth?
- Personal code?
- Some biometrics?
- A combination of these?
- ...

Identifying myself

What is "me"?

- Name?
- Date of birth?
- Personal code?
- Some biometrics?
- A combination of these?
- . . .

What is in the protocol?

- My public key,
- . . . bound to me by a certificate
- . . . that is signed by an authority
- . . . that the relying party trusts

Identifying myself

What is "me"?

- Name?
- Date of birth?
- Personal code?
- Some biometrics?
- A combination of these?
- ...

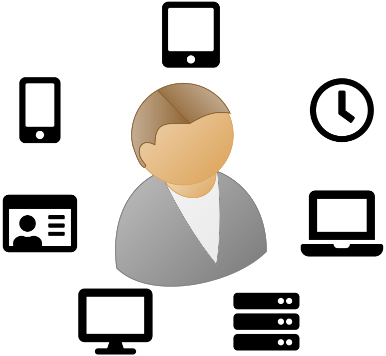
What is in the protocol?

- My public key,
- ... bound to me by a certificate
- ... that is signed by an authority
- ... that the relying party trusts

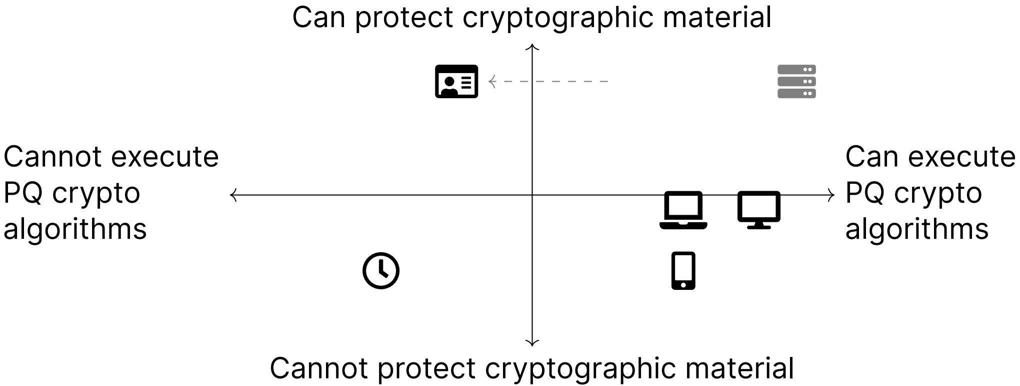
How do I use my public key?

- I have the corresponding private key.
- I do cryptographic operations with it.

Me and computing devices

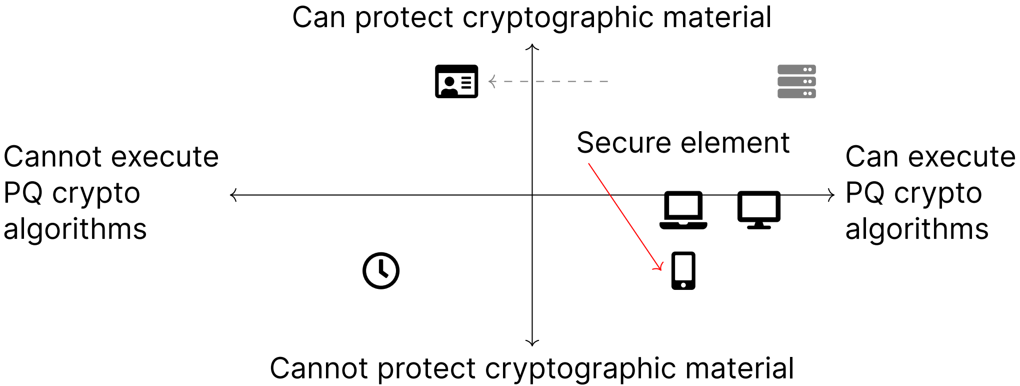


Devices



Can protect cryptographic material
Cannot execute PQ crypto algorithms
Can execute PQ crypto algorithms
Cannot protect cryptographic material

Devices



Can protect cryptographic material
Secure element
Cannot execute PQ crypto algorithms
Can execute PQ crypto algorithms
Cannot protect cryptographic material

What devices running authentication protocols are there?

For user

- Computers
- Smartphones
- Secure elements
- Embedded devices
- Smartcards, tokens, dongles, . . .

For relying party

- Computers (servers)
- Hardware Security Modules (HSM)

- These need to compute digital signatures

What devices running authentication protocols are there?

For user

- Computers
- Smartphones
- Secure elements
- Embedded devices
- Smartcards, tokens, dongles, . . .
- Threshold cryptography

For relying party

- Computers (servers)
- Hardware Security Modules (HSM)

- These need to compute digital signatures

Threshold signing

- Parties $P_1, \ldots, P_n$ hold shares $sk_1, \ldots, sk_n$ of the signing key
- The corresponding public key pk is known to everyone
- Given a message M, parties $P_1, \ldots, P_n$ can run a protocol and produce a signature σ
- Using pk, anyone can verify that σ is a signature on M
- This setting is particularly interesting when the verification procedure is a “standard” one

Support for PQ authentication

Smartcards

- Signing has been realized
  - … for Dilithium, Falcon
  - But no side-channel protection
- Side-channel protection requires too much memory
- Falcon's floating-point operations are hard to protect

Threshold cryptography

- There exist threshold signing protocols for Dilithium and Falcon
  - Too inefficient for Smart-ID
- Dilithium has inspired more efficient threshold schemes
  - Verification algorithm is different
- (Thresholdizing hash-based constructions is harder)

Key encapsulation may be simpler

TOPCOAT — our Dilithium-inspired threshold signature scheme

- Designed specifically for two signing parties
- Applies Dilithium’s compression techniques for public keys and signatures
- Relies on same lattice-based hardness assumptions as Dilithium
  - … with equally efficient security reductions
- Efficient in practice

TOPCOAT — our Dilithium-inspired threshold signature scheme

- Designed specifically for two signing parties
- Applies Dilithium's compression techniques for public keys and signatures
- Relies on same lattice-based hardness assumptions as Dilithium
  - ... with equally efficient security reductions
- Efficient in practice

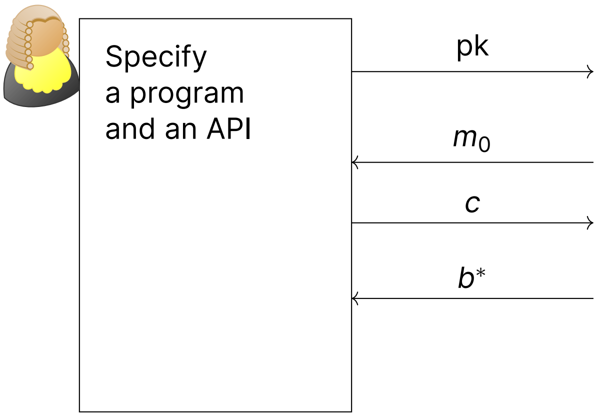
Everything OK with PQ authentication?

In practice, yes. In theory, ...

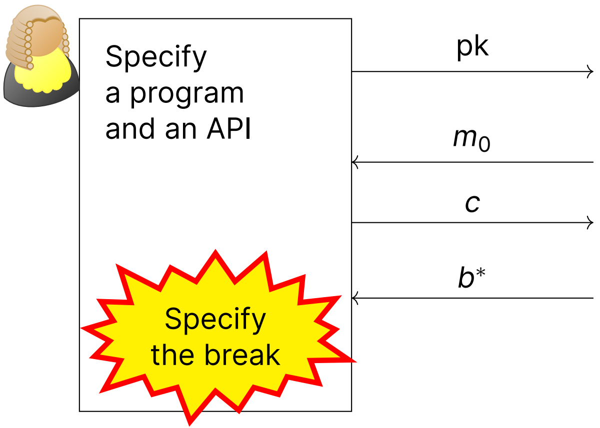
On details of security proofs

- There are (computational) hardness assumptions
  - E.g. factoring products of large primes, finding discrete logarithms, finding short vectors in lattices, learning with errors
- There are cryptographic primitives
  - E.g. digital signature, public-key encryption, hash function
- There a security definitions of these primitives
  - E.g. existential unforgeability under chosen-message attacks, collision-resistance

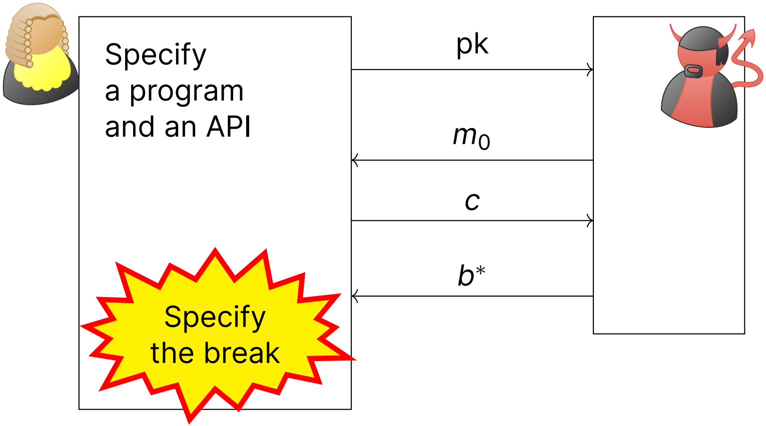
Hardness assumptions and security definitions



Hardness assumptions and security definitions



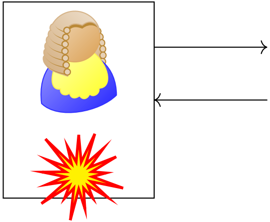
Hardness assumptions and security definitions



Postulate that

there is no 
that causes the break

Reductions

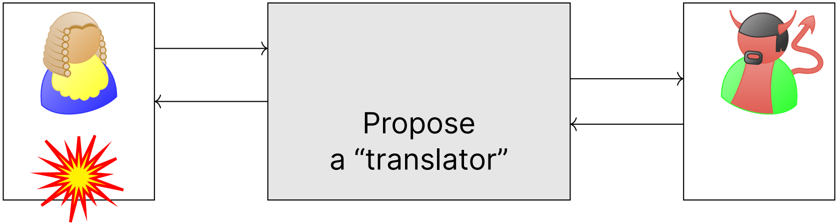


If a computational problem is hard then a cryptographic primitive is secure

Reductions

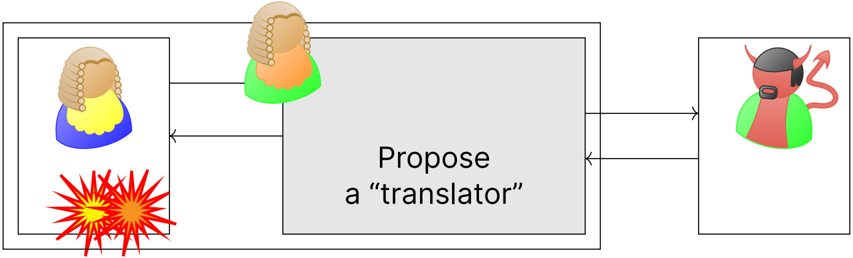
If a computational problem is hard then a cryptographic primitive is secure

Reductions



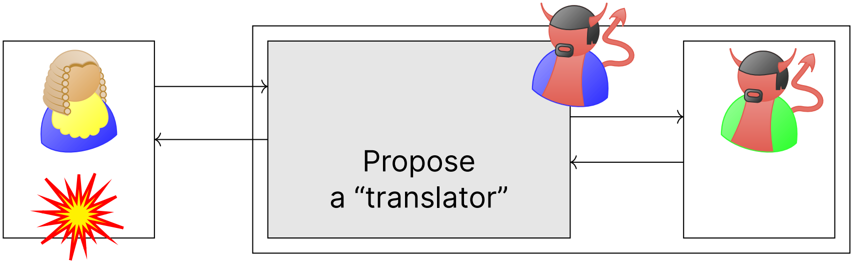
If a computational problem is hard then a cryptographic primitive is secure

Reductions



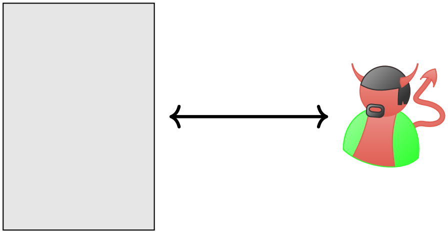
If a computational problem is hard then a cryptographic primitive is secure

Reductions

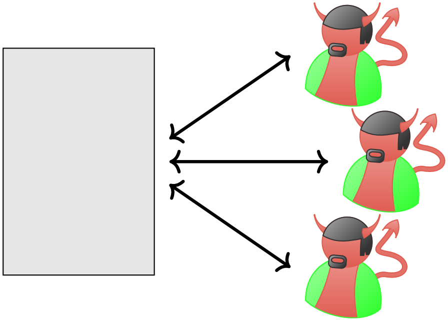


If a computational problem is hard then a cryptographic primitive is secure

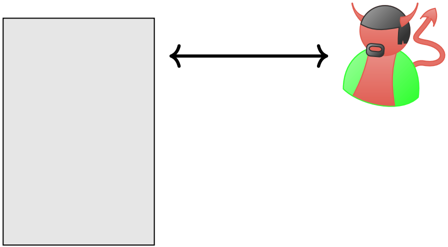
How reductions may use



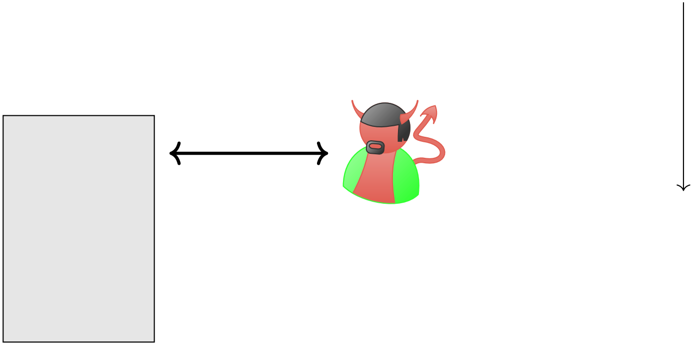
How reductions may use 



How reductions may use 

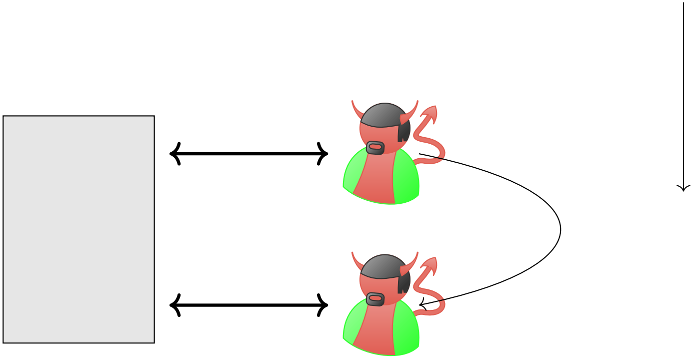


How reductions may use 



Execute

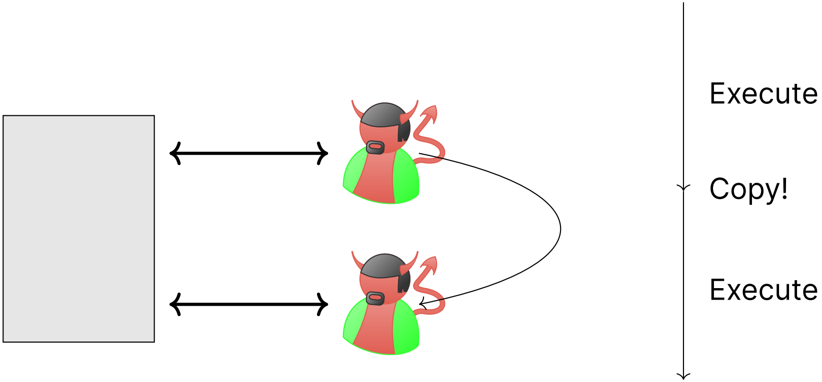
How reductions may use 



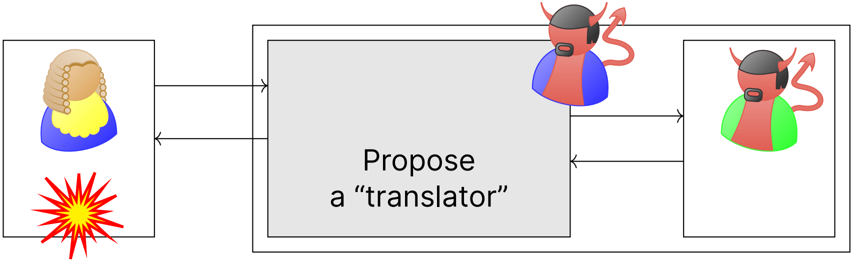
Execute

Copy!

How reductions may use 

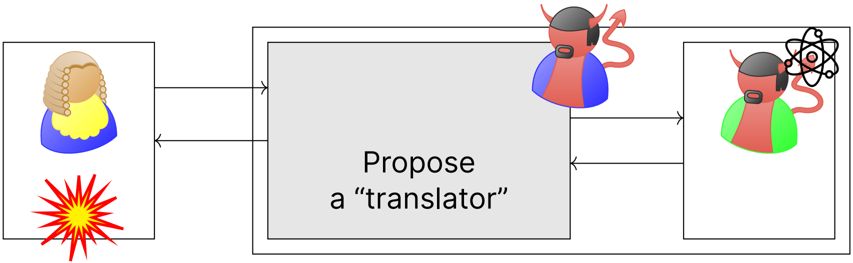


Quantum reductions



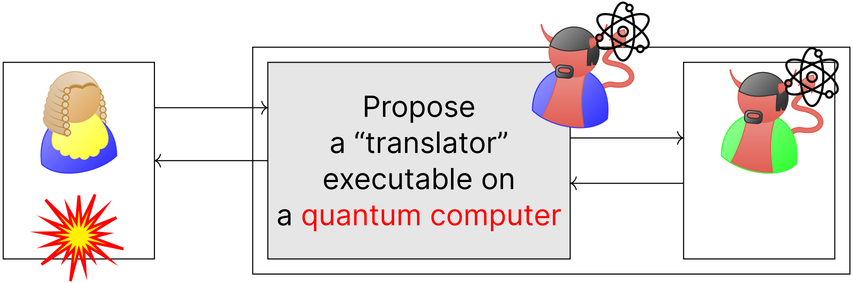
If a computational problem is hard then a cryptographic primitive is secure

Quantum reductions



If a computational problem is hard then a cryptographic primitive is secure

Quantum reductions



If a computational problem is hard then a cryptographic primitive is secure

“Plausibly quantum-secure” constructions

- Some hardness assumptions do not hold in presence of quantum adversaries. Others do
  - A hardness assumption holds in presence of quantum adversaries $\Rightarrow$ it holds in presence of only classical adversaries
- Dilithium, Falcon, etc. have proofs that quantum-reduce their security to some quantum-valid hardness assumption
- Many other constructions have proofs that only classically reduce their security to some quantum-valid hardness assumption
  - This includes TOPCOAT and other threshold lattice-based signatures

Way forward

- Certain steps cannot be executed on a quantum computer
  - Rewinding
  - Gradually defining a function
    - ... that is meant to be executed in quantum superposition
  - Enumerating the queries to such a function
- There are theorems stating that under certain conditions, these steps in reductions can still be emulated by a quantum computer
  - These conditions do not appear too onerous
- There is hope that we get a quantum reduction for TOPCOAT

Authentication vs. signing

Authentication

- Takes place in a single moment of time

Signing and verification

- There is a significant window of time b/w signing and verification

Authentication vs. signing

Authentication

- Takes place in a single moment of time
- Takes place between a fixed set of (two) parties

Signing and verification

- There is a significant window of time b/w signing and verification
- The set of relevant parties is not fixed during signing

Authentication vs. signing

Authentication

- Takes place in a single moment of time
- Takes place between a fixed set of (two) parties

  ⇒ May use any “exotic” technology that these parties support

Signing and verification

- There is a significant window of time b/w signing and verification
- The set of relevant parties is not fixed during signing

  ⇒ The more standardized the technology, the better

Authentication vs. signing

Authentication

- Takes place in a single moment of time
- Takes place between a fixed set of (two) parties

⇒ May use any “exotic” technology that these parties support

Signing and verification

- There is a significant window of time b/w signing and verification
- The set of relevant parties is not fixed during signing

⇒ The more standardized the technology, the better

Commonality

“Digital signature” cryptographic primitive tends to be important for both*

Post-quantum digital signing?

Can threshold cryptography help?

- As we said, existing threshold protocols for Dilithium are too inefficient
- But these protocols are “generic”. E.g. for any number of signers
  - Also, existing implementations are “generic”
- If we try to take advantage of the details of our setting, can we overcome the inefficiencies?
  - Number of parties is 2
  - The algorithm to be implemented is Dilithium
    - Optimize the components and their compositions

Secure multiparty computation (SMC)

- There's a function $(Y_1, \ldots, Y_n) = f(X_1, \ldots, X_n)$ (may be randomized)
- There are n parties $P_1, \ldots, P_n$. Party P_i has a value x_i
- There is an *access structure* $\mathcal{A} \subseteq 2^{\{1,\ldots,n\}}$ if $n = 2$, then $\mathcal{A} = \{\{1, 2\}\}$

Secure multiparty computation (SMC)

- There's a function $(Y_1, \ldots, Y_n) = f(X_1, \ldots, X_n)$ (may be randomized)
- There are n parties $P_1, \ldots, P_n$. Party P_i has a value x_i
- There is an *access structure* $\mathcal{A} \subseteq 2^{\{1,\ldots,n\}}$ if $n = 2$, then $\mathcal{A} = \{\{1, 2\}\}$
- A secure MPC protocol Π for f does the following:
  - $\forall i$: P_i inputs x_i to the protocol
  - $\forall i$: P_i obtains y_i, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$

Secure multiparty computation (SMC)

- There's a function $(Y_1, \ldots, Y_n) = f(X_1, \ldots, X_n)$ (may be randomized)
- There are n parties $P_1, \ldots, P_n$. Party P_i has a value x_i
- There is an *access structure* $\mathcal{A} \subseteq 2^{\{1,\ldots,n\}}$ if $n = 2$, then $\mathcal{A} = \{\{1, 2\}\}$
- A secure MPC protocol Π for f does the following:
  - $\forall i$: P_i inputs x_i to the protocol
  - $\forall i$: P_i obtains y_i, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$
  - $\forall \mathbf{N} \subseteq \{1, \ldots, n\}$: if $\mathbf{N} \notin \mathcal{A}$, then the group $\{P_i\}_{i\in\mathbf{N}}$ learns nothing beyond $\{x_i\}_{i\in\mathbf{N}}, \{y_i\}_{i\in\mathbf{N}}$

Secure multiparty computation (SMC)

- There's a function $(Y_1, \ldots, Y_n) = f(X_1, \ldots, X_n)$ (may be randomized)
- There are n parties $P_1, \ldots, P_n$. Party P_i has a value x_i
- There is an *access structure* $\mathcal{A} \subseteq 2^{\{1,\ldots,n\}}$ if $n = 2$, then $\mathcal{A} = \{\{1, 2\}\}$
- A secure MPC protocol Π for f does the following:
  - $\forall i$: P_i inputs x_i to the protocol
  - $\forall i$: P_i obtains y_i, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$
  - $\forall \mathbf{N} \subseteq \{1, \ldots, n\}$: if $\mathbf{N} \notin \mathcal{A}$, then the group $\{P_i\}_{i \in \mathbf{N}}$ learns nothing beyond $\{x_i\}_{i \in \mathbf{N}}, \{y_i\}_{i \in \mathbf{N}}$
- Π may provide security against *passive* or *active* corruptions
  - Kinds of active security: fail-stop, identifiable abort, . . . , full security

A common technique of SMC

- A private value v is **additively shared**:
  - Fix a modulus N of suitable size
  - Party P_i holds $v_i \in \{0, \ldots, N-1\}$, s.t. $v_1 + \cdots + v_n = v \pmod{N}$
    - (there is more, if we want security against active adversaries)

A common technique of SMC

- A private value v is **additively shared**:
  - Fix a modulus N of suitable size
  - Party P_i holds $v_i \in \{0, \ldots, N-1\}$, s.t. $v_1 + \cdots + v_n = v \pmod{N}$
    - (there is more, if we want security against active adversaries)
- Addition (modulo N) of private values requires no communication
- There are protocols for other operations with private values
  - These are composed to a protocol for f

Typical performance profile of SMC protocols

- Additions (and linear combinations) are “free”
- Revealing a value takes some communication
  - Entering a value may be free, or take some communication, too
- Multiplications take more time / communication
- Equality checks take even more time
- Inequality checks take even (a lot) more time

Off-/online constructions of SMC protocols

- Compute some “correlated” randomness beforehand, such that computations are faster once the inputs to f are available

Off-/online constructions of SMC protocols

- Compute some “correlated” randomness beforehand, such that computations are faster once the inputs to f are available
  - E.g. **multiplication triple:** P_i holds random $a_i, b_i, c_i \in \{0, \ldots, N-1\}$
  - Correlated: $(a_1 + \cdots + a_n) \cdot (b_1 + \cdots + b_n) = (c_1 + \cdots + c_n)$
  - With its help, multiplication reduces to linear operations and openings.

Off-/online constructions of SMC protocols

- Compute some "correlated" randomness beforehand, such that computations are faster once the inputs to f are available
  - E.g. **multiplication triple:** P_i holds random $a_i, b_i, c_i \in \{0, \dots, N-1\}$
  - Correlated: $(a_1 + \cdots + a_n) \cdot (b_1 + \cdots + b_n) = (c_1 + \cdots + c_n)$
  - With its help, multiplication reduces to linear operations and openings.
- $P_1, \dots, P_n$ may use more heavyweight protocols to generate correlated randomness
  - This may happen during "downtime"

Off-/online constructions of SMC protocols

- Compute some “correlated” randomness beforehand, such that computations are faster once the inputs to f are available
  - E.g. **multiplication triple:** P_i holds random $a_i, b_i, c_i \in \{0, \ldots, N-1\}$
  - Correlated: $(a_1 + \cdots + a_n) \cdot (b_1 + \cdots + b_n) = (c_1 + \cdots + c_n)$
  - With its help, multiplication reduces to linear operations and openings.
- $P_1, \ldots, P_n$ may use more heavyweight protocols to generate correlated randomness
  - This may happen during “downtime”
- Or, there may be some extra “trusted” party that generates correlated randomness

Correlated randomness for (in)equalities

- Faster protocols for inequalities are currently an active research area
- Involve novel forms of correlated randomness
- Proposals are often secure only against passive adversaries

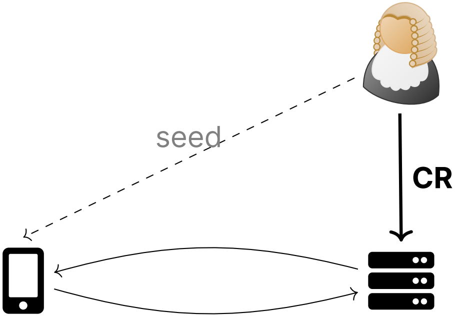
Our in-progress work

- Three parties: Phone, Server, Correlated Randomness Generator (CRG)
- Security against one of the parties:
  - against actively corrupted Phone, or
  - against passively corrupted Server
    - Appears to give **privacy** against actively corrupted Server
    - I believe we can turn this to **security** against actively corrupted Server
  - (against CRG, when Phone and Server are honest)

Our in-progress work

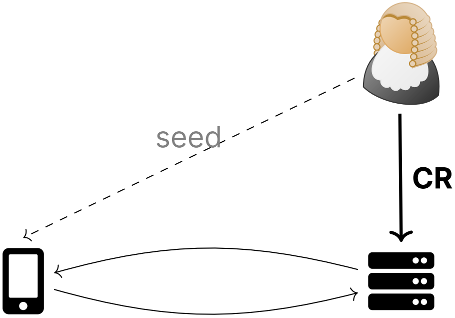
- Three parties: Phone, Server, Correlated Randomness Generator (CRG)
- Security against one of the parties:
  - against actively corrupted Phone, or
  - against passively corrupted Server
    - Appears to give **privacy** against actively corrupted Server
    - I believe we can turn this to **security** against actively corrupted Server
  - (against CRG, when Phone and Server are honest)
- ca. 20 rounds of communication
- Hundreds of KB of exchanged messages
- Megabytes of correlated randomness going to Server

Deployment?



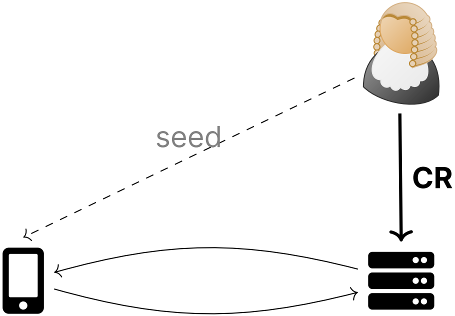
- Can Server and CRG both be run by a TSP?
  - The same TSP?
- Isolation between Server and CRG must be good
- CRG could be deployed offline or online

Deployment?



- Can Server and CRG both be run by a TSP?
  - The same TSP?
- Isolation between Server and CRG must be good
- CRG could be deployed offline or online
- We are going to obtain experience running the CRG under hardware isolation

Deployment?

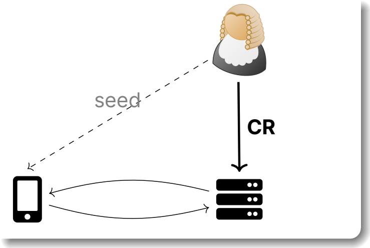


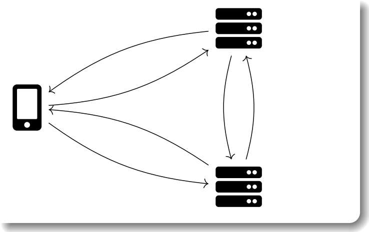
- Can Server and CRG both be run by a TSP?
  - The same TSP?
- Isolation between Server and CRG must be good
- CRG could be deployed offline or online
- We are going to obtain experience running the CRG under hardware isolation

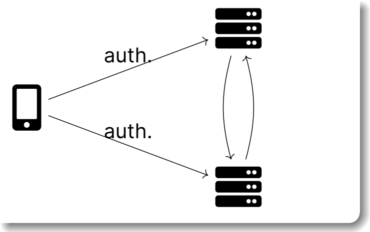
Performance?

Will it be acceptable?

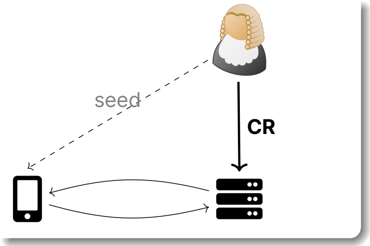
Alternative 3-party, 1-corruption deployments



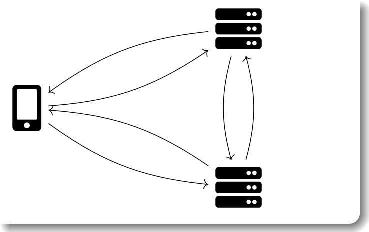




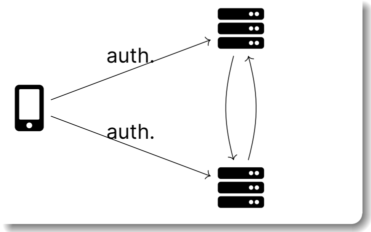
Alternative 3-party, 1-corruption deployments



✅ Easiest to deploy



✅ Simplest protocol



✅ Simplest for phone

❌ No keyshare in phone

Conclusions?

Excellent, but not hopeless…