Private Set Intersection for Small Sets

Mike Rosulek, Oregon State University

NIST WPEC, September 24, 2024

Joint work with: Yeongjin Jang, Stanislav Lyakhov, Lawrence Roy, Ni Trieu

what is private set intersection (PSI)?

Alice

p x o

n r e

s u m

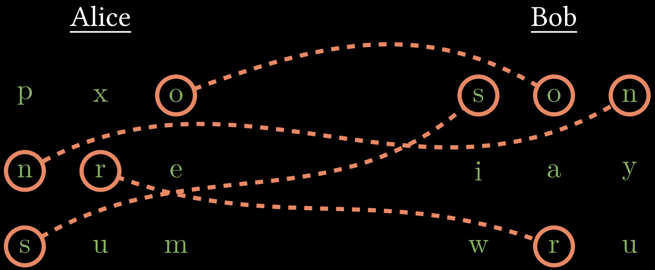
Bob

s o n

i a y

w r u

what is private set intersection (PSI)?



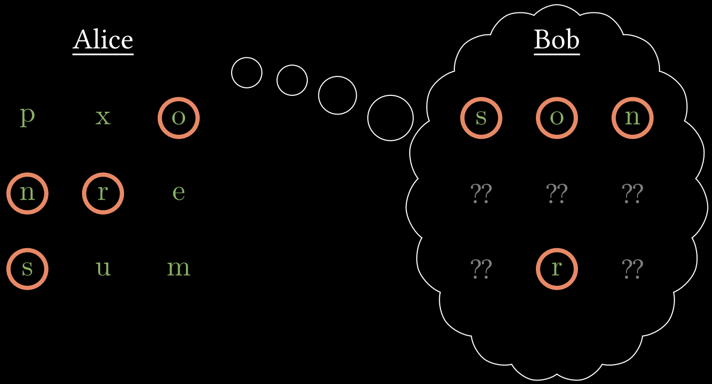
Alice

p x o
n r e
s u m

Bob

s o n
i a y
w r u

what is private set intersection (PSI)?



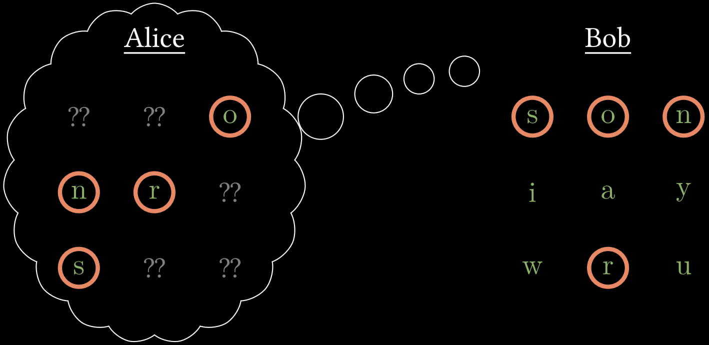
Alice

p x o
n r e
s u m

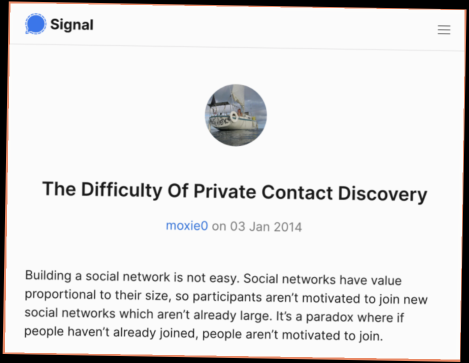
Bob

s o n
?? ?? ??
?? r ??

what is private set intersection (PSI)?



why use PSI?



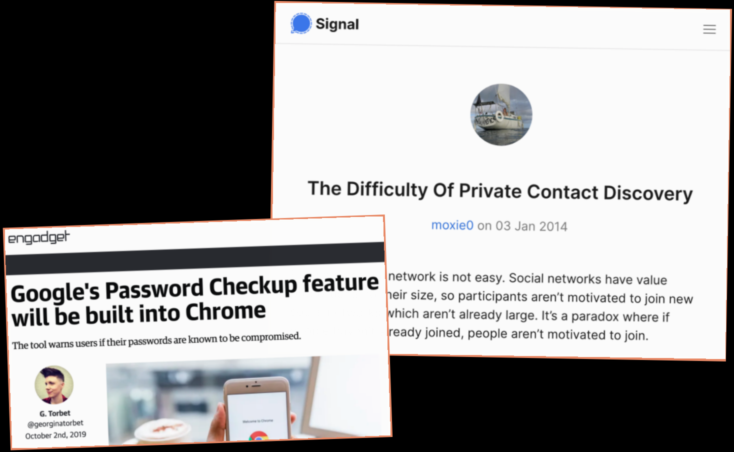
The Difficulty Of Private Contact Discovery

moxie0 on 03 Jan 2014

Building a social network is not easy. Social networks have value proportional to their size, so participants aren't motivated to join new social networks which aren't already large. It's a paradox where if people haven't already joined, people aren't motivated to join.

{my phone contacts} ∩ {users of your service}

why use PSI?



Signal

The Difficulty Of Private Contact Discovery

moxie0 on 03 Jan 2014

network is not easy. Social networks have value
heir size, so participants aren't motivated to join new
which aren't already large. It's a paradox where if
ready joined, people aren't motivated to join.

engadget

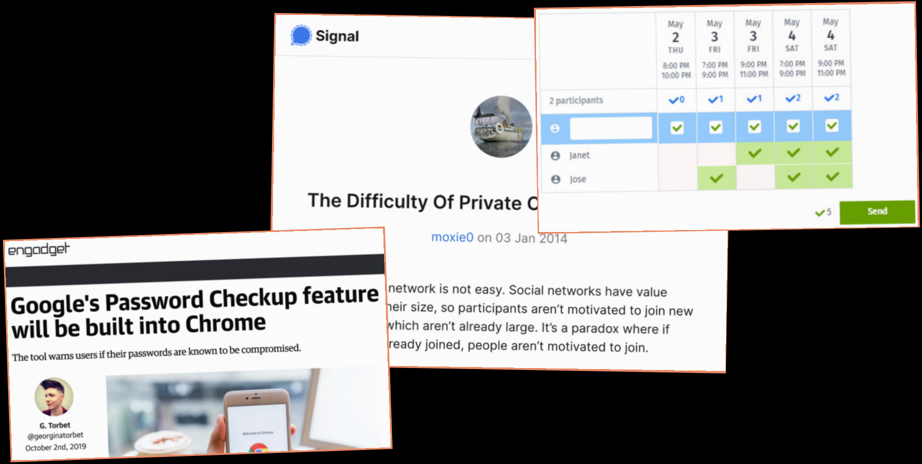
Google's Password Checkup feature will be built into Chrome

The tool warns users if their passwords are known to be compromised.

G. Torbet
@georginatorbet
October 2nd, 2019

{my passwords} ∩ {passwords found in breaches}

why use PSI?



Signal

The Difficulty Of Private C

moxie0 on 03 Jan 2014

network is not easy. Social networks have value
heir size, so participants aren't motivated to join new
which aren't already large. It's a paradox where if
ready joined, people aren't motivated to join.

	May 2 THU 8:00 PM 10:00 PM	May 3 FRI 7:00 PM 9:00 PM	May 3 FRI 9:00 PM 11:00 PM	May 4 SAT 7:00 PM 9:00 PM	May 4 SAT 9:00 PM 11:00 PM
2 participants	✓0	✓1	✓1	✓2	✓2
	☑	☑	☑	☑	☑
Janet			✓	✓	✓
Jose		✓		✓	✓

✓5 Send

engadget

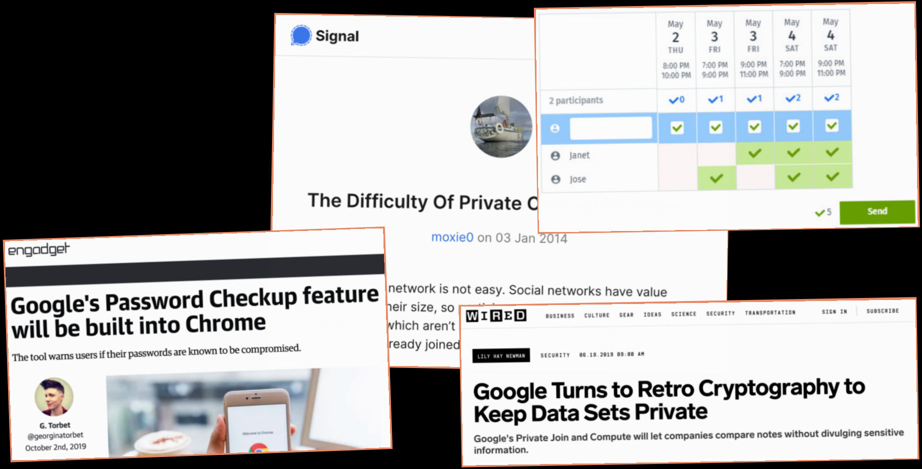
Google's Password Checkup feature will be built into Chrome

The tool warns users if their passwords are known to be compromised.

G. Torbet
@georginatorbet
October 2nd, 2019

{my availability} ∩ {your availability}

why use PSI?



The Difficulty Of Private C

moxie0 on 03 Jan 2014

network is not easy. Social networks have value
heir size, so
which aren't
ready joined

	May 2 THU 8:00 PM 10:00 PM	May 3 FRI 7:00 PM 9:00 PM	May 3 FRI 9:00 PM 11:00 PM	May 4 SAT 7:00 PM 9:00 PM	May 4 SAT 9:00 PM 11:00 PM
2 participants	✓0	✓1	✓1	✓2	✓2
	☑	☑	☑	☑	☑
Janet			✓	✓	✓
Jose		✓		✓	✓

✓5

Google's Password Checkup feature will be built into Chrome

The tool warns users if their passwords are known to be compromised.

G. Torbet
@georginatorbet
October 2nd, 2019

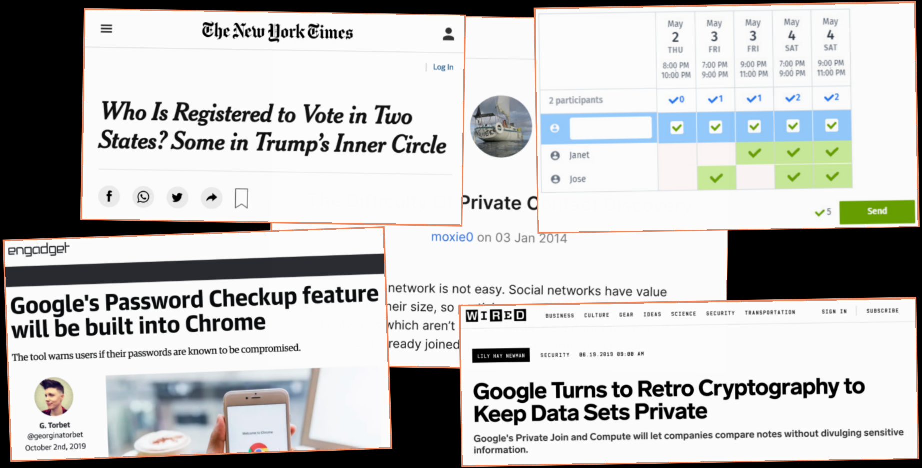
LILY HAY NEWMAN SECURITY 06.19.2019 09:00 AM

Google Turns to Retro Cryptography to Keep Data Sets Private

Google's Private Join and Compute will let companies compare notes without divulging sensitive information.

{people who saw ad} ∩ {customers who made purchases}

why use PSI?



The New York Times

Log In

Who Is Registered to Vote in Two States? Some in Trump's Inner Circle

Private C

moxie0 on 03 Jan 2014

network is not easy. Social networks have value
heir size, so
which aren't
ready joined

	May 2 THU 8:00 PM 10:00 PM	May 3 FRI 7:00 PM 9:00 PM	May 3 FRI 9:00 PM 11:00 PM	May 4 SAT 7:00 PM 9:00 PM	May 4 SAT 9:00 PM 11:00 PM
2 participants	✓0	✓1	✓1	✓2	✓2
	☑	☑	☑	☑	☑
Janet			✓	✓	✓
Jose		✓		✓	✓

✓5 Send

engadget

Google's Password Checkup feature will be built into Chrome

The tool warns users if their passwords are known to be compromised.

G. Torbet
@georginatorbet
October 2nd, 2019

WIRED BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY TRANSPORTATION SIGN IN SUBSCRIBE

LILY HAY NEWMAN SECURITY 06.19.2019 09:00 AM

Google Turns to Retro Cryptography to Keep Data Sets Private

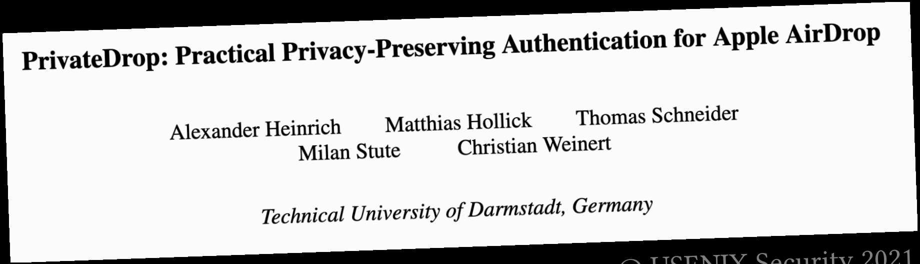
Google's Private Join and Compute will let companies compare notes without divulging sensitive information.

{voters registered in OR} ∩ {voters registered in NY}

motivating PSI for small sets

PSI for small sets = PSI for *personal privacy*

motivating PSI for small sets



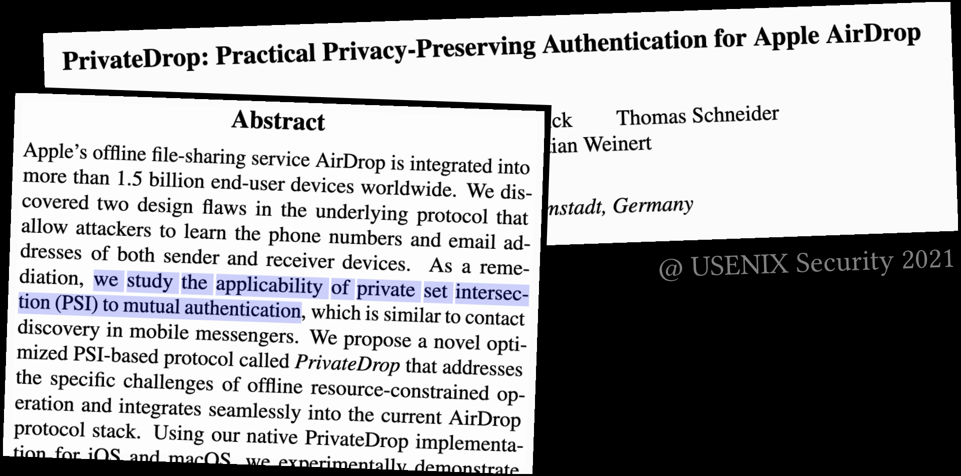
PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop

Alexander Heinrich Matthias Hollick Thomas Schneider
Milan Stute Christian Weinert

Technical University of Darmstadt, Germany

@ USENIX Security 2021

motivating PSI for small sets



PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop

ck Thomas Schneider
ian Weinert

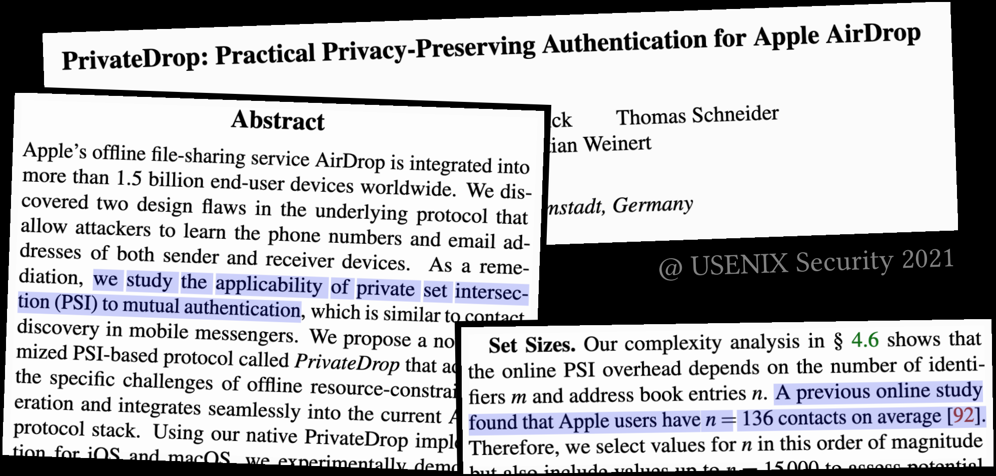
nstadt, Germany

Abstract

Apple's offline file-sharing service AirDrop is integrated into more than 1.5 billion end-user devices worldwide. We discovered two design flaws in the underlying protocol that allow attackers to learn the phone numbers and email addresses of both sender and receiver devices. As a remediation, we study the applicability of private set intersection (PSI) to mutual authentication, which is similar to contact discovery in mobile messengers. We propose a novel optimized PSI-based protocol called *PrivateDrop* that addresses the specific challenges of offline resource-constrained operation and integrates seamlessly into the current AirDrop protocol stack. Using our native PrivateDrop implementation for iOS and macOS, we experimentally demonstrate

@ USENIX Security 2021

motivating PSI *for small sets*



PrivateDrop: Practical Privacy-Preserving Authentication for Apple AirDrop

ck Thomas Schneider
ian Weinert

nstadt, Germany

@ USENIX Security 2021

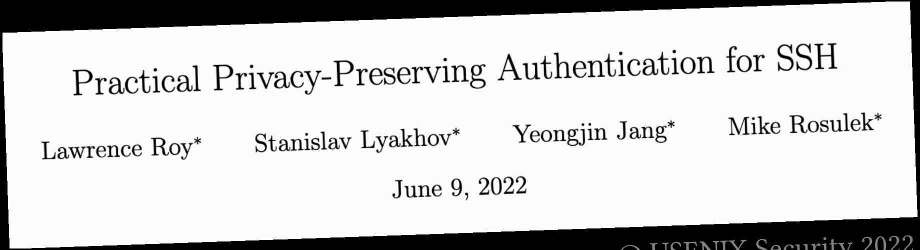
Abstract

Apple's offline file-sharing service AirDrop is integrated into more than 1.5 billion end-user devices worldwide. We discovered two design flaws in the underlying protocol that allow attackers to learn the phone numbers and email addresses of both sender and receiver devices. As a remediation, we study the applicability of private set intersection (PSI) to mutual authentication, which is similar to contact discovery in mobile messengers. We propose a no
mized PSI-based protocol called *PrivateDrop* that ac
the specific challenges of offline resource-constrai
eration and integrates seamlessly into the current A
protocol stack. Using our native PrivateDrop impl
tion for iOS and macOS, we experimentally demo

Set Sizes. Our complexity analysis in § 4.6 shows that the online PSI overhead depends on the number of identifiers *m* and address book entries *n*. A previous online study found that Apple users have $n = 136$ contacts on average [92]. Therefore, we select values for *n* in this order of magnitude but also include values up to $n = 15\,000$ to assess potential

[92] = Stute, Narain, Mariotto, Heinrich, Kreitschmann, Noubir, Hollick @ *USENIX* 2019

motivating PSI for small sets

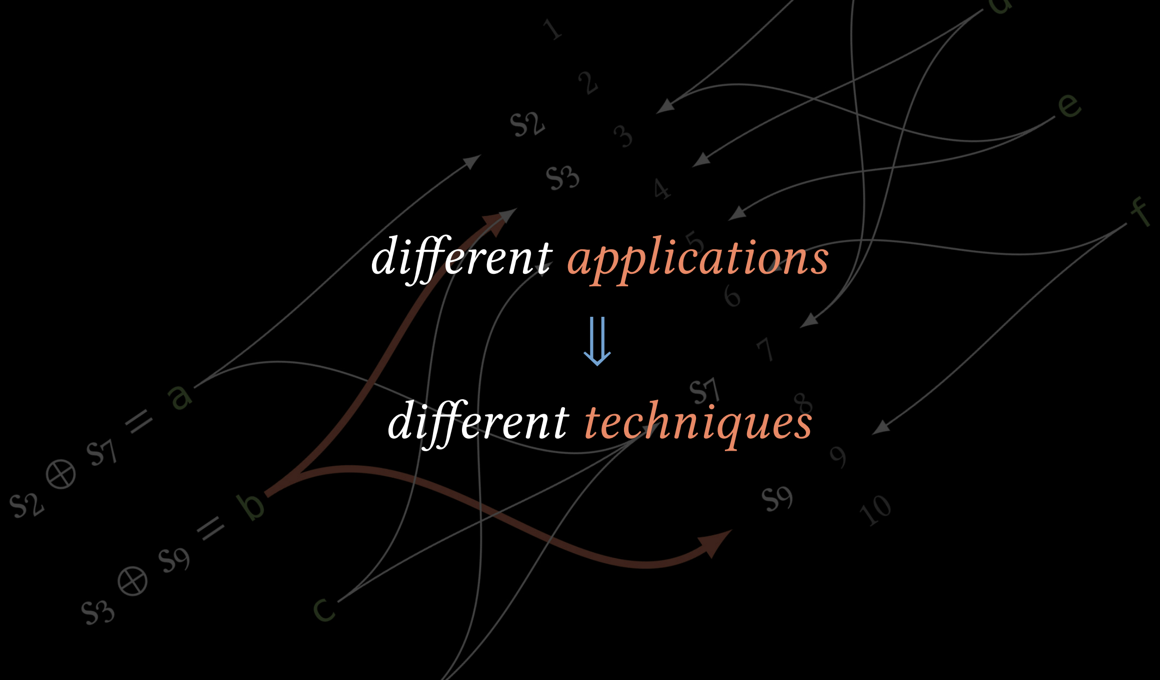


Practical Privacy-Preserving Authentication for SSH

Lawrence Roy* Stanislav Lyakhov* Yeongjin Jang* Mike Rosulek*

June 9, 2022

@ USENIX Security 2022



different applications

⇓

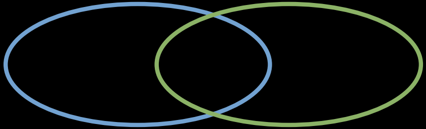
different techniques

PSI on **small sets** (hundreds)

- private availability poll
- key agreement techniques

PSI on **small sets** (hundreds)

- private availability poll
- key agreement techniques

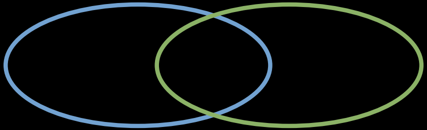


PSI on **large sets** (millions)

- double-registered voters
- OT extension; combinatorial tricks

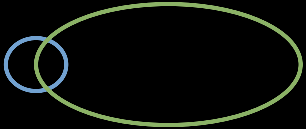
PSI on **small sets** (hundreds)

- private availability poll
- key agreement techniques



PSI on **large sets** (millions)

- double-registered voters
- OT extension; combinatorial tricks

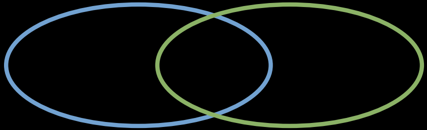


PSI on **asymmetric sets** (100 : billion)

- contact discovery; password checkup
- offline phase; leakage

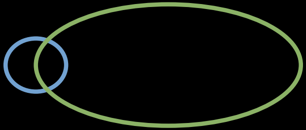
PSI on **small sets** (hundreds)

- private availability poll
- key agreement techniques



PSI on **large sets** (millions)

- double-registered voters
- OT extension; combinatorial tricks



PSI on **asymmetric sets** (100 : billion)

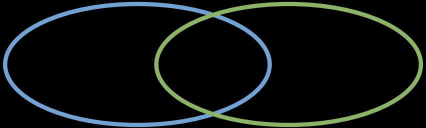
- contact discovery; password checkup
- offline phase; leakage

computing on the intersection

- sales statistics about intersection
- generic MPC

PSI on **small sets** (hundreds)

- private availability poll
- key agreement techniques

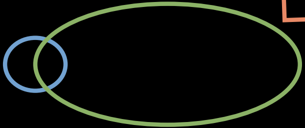


PSI on **large sets** (millions)

- oters
- inatorial tricks

Not to mention:

- approximate/fuzzy matching
- more than 2 parties/sets
- private set *union*



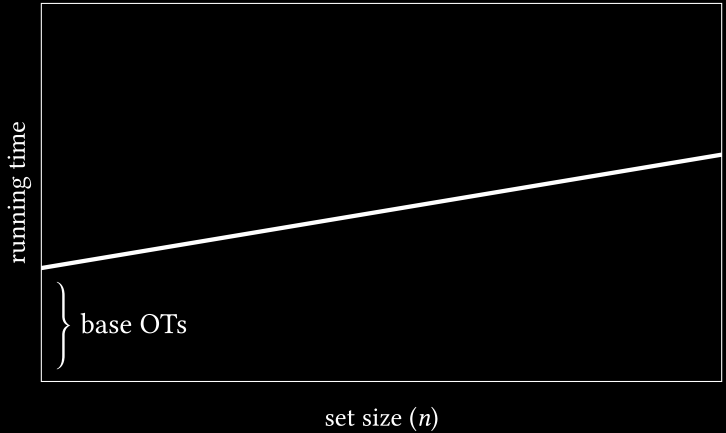
PSI on **asymmetric sets** (100 : billion)

- contact discovery; password checkup
- offline phase; leakage

computing on the intersection

- sales statistics about intersection
- generic MPC

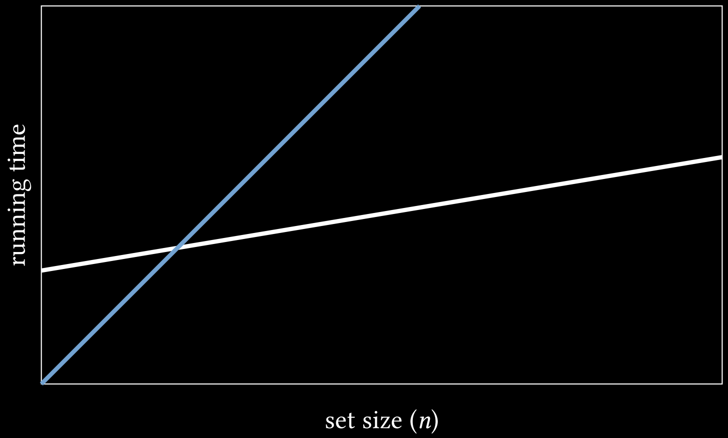
PSI techniques for small sets



OT-based PSI:

- 128 base OTs
- $O(n)$ symm-key ops

PSI techniques for small sets



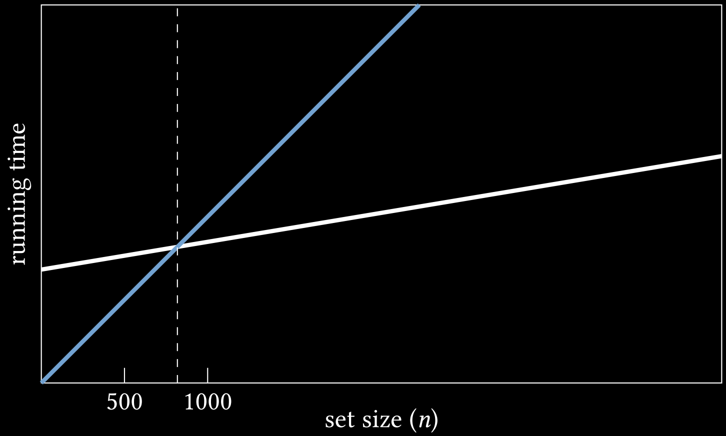
OT-based PSI:

- 128 base OTs
- $O(n)$ symm-key ops

KA-based PSI:

- $O(n)$ pub-key ops

PSI techniques for small sets



OT-based PSI:

- 128 base OTs
- $O(n)$ symm-key ops

KA-based PSI:

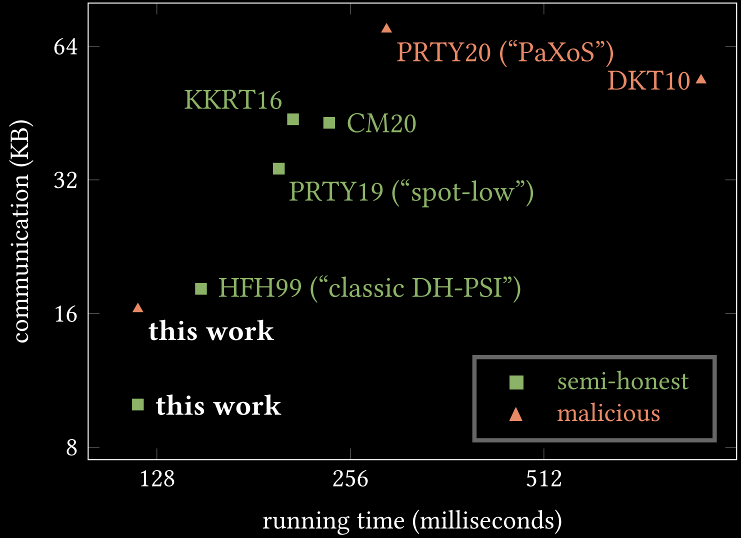
- $O(n)$ pub-key ops

Compact and Malicious *Private Set Intersection* for Small Sets

Mike Rosulek, Oregon State University
Ni Trieu, Arizona State University

appeared at ACM CCS 2021

PSI cost: 256 items per party:



our **semi-honest** PSI:

- 45% ↓ communication
- 20% ↓ runtime

our **malicious** PSI:

- 10% ↓ communication
- 20% ↓ runtime

vs. best *semi-honest* PSI!

- 75% ↓ communication
- 55% ↓ runtime

vs. best malicious PSI

Alice (random oracle $H : \{0,1\}^* \to \mathbb{G}$) **Bob**

$x_1, x_2, \ldots$

$y_1, y_2, \ldots$

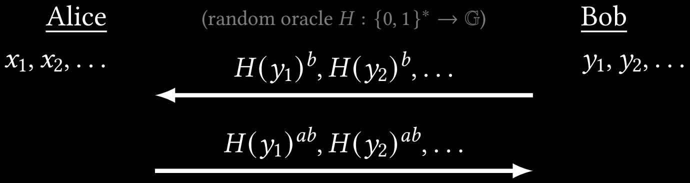
[HubermanFranklinHogg99]

Alice (random oracle $H : \{0,1\}^* \longrightarrow \mathbb{G}$) **Bob**

$x_1, x_2, \ldots$ $y_1, y_2, \ldots$

$$\xleftarrow{\quad H(y_1)^b, H(y_2)^b, \ldots \quad}$$

[HubermanFranklinHogg99]



Alice

(random oracle $H : \{0,1\}^* \rightarrow \mathbb{G}$)

Bob

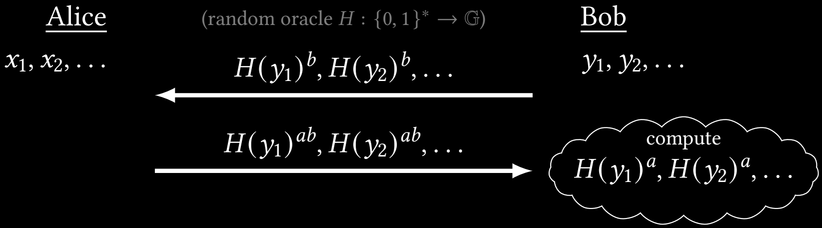
$x_1, x_2, \ldots$

$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$ ←

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$ →

[HubermanFranklinHogg99]



Alice

(random oracle $H : \{0,1\}^* \to \mathbb{G}$)

Bob

$x_1, x_2, \ldots$

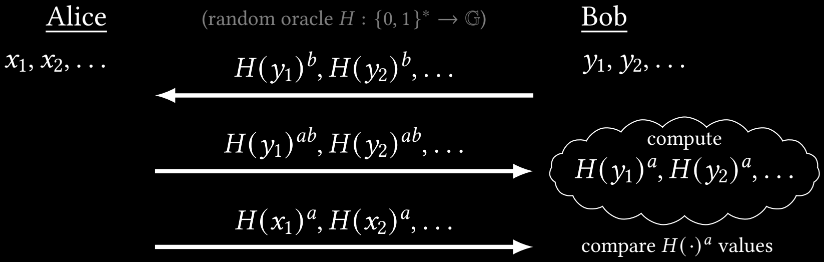
$H(y_1)^b, H(y_2)^b, \ldots$

$y_1, y_2, \ldots$

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

compute
$H(y_1)^a, H(y_2)^a, \ldots$

[HubermanFranklinHogg99]



Alice

(random oracle $H : \{0,1\}^* \to \mathbb{G}$)

Bob

$x_1, x_2, \ldots$

$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

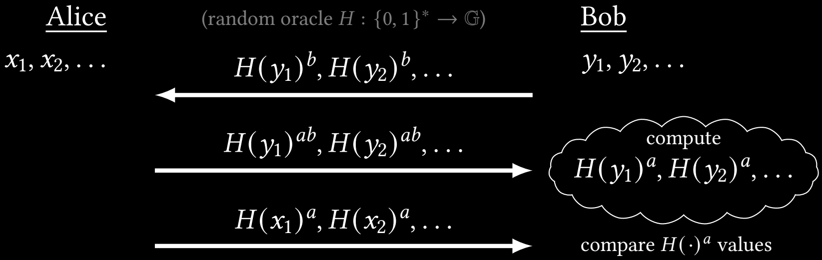
$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

compute $H(y_1)^a, H(y_2)^a, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

compare $H(\cdot)^a$ values

[HubermanFranklinHogg99]



Alice (random oracle $H : \{0,1\}^* \to \mathbb{G}$) **Bob**

$x_1, x_2, \ldots$

$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

compute $H(y_1)^a, H(y_2)^a, \ldots$

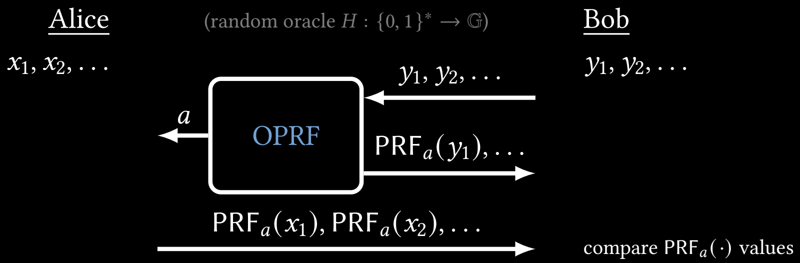
$H(x_1)^a, H(x_2)^a, \ldots$

compare $H(\cdot)^a$ values

Semi-honest security:

- $x \mapsto H(x)^a$ is a PRF (DDH assumption + random oracle)
- first two messages are an oblivious PRF protocol

[HubermanFranklinHogg99]



Alice (random oracle $H : \{0,1\}^* \to \mathbb{G}$) Bob

$x_1, x_2, \ldots$

$y_1, y_2, \ldots$

OPRF

a

$y_1, y_2, \ldots$

$\mathsf{PRF}_a(y_1), \ldots$

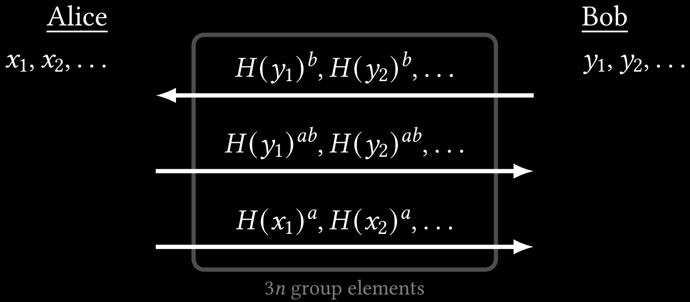
$\mathsf{PRF}_a(x_1), \mathsf{PRF}_a(x_2), \ldots$

compare $\mathsf{PRF}_a(\cdot)$ values

Semi-honest security:

- $x \mapsto H(x)^a$ is a PRF (DDH assumption + random oracle)
- first two messages are an oblivious PRF protocol
- standard OPRF→PSI paradigm [FreedmanIshaiPinkasReingold05]

[HubermanFranklinHogg99]



Alice

$x_1, x_2, \ldots$

Bob

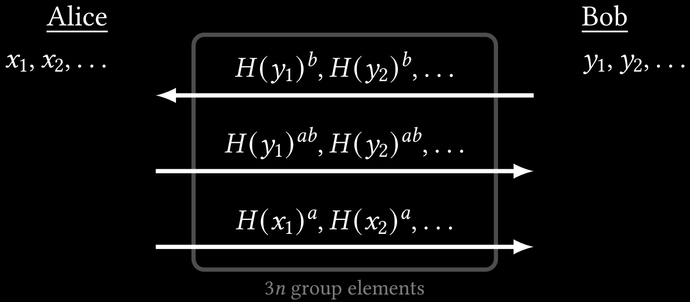
$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

$3n$ group elements



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

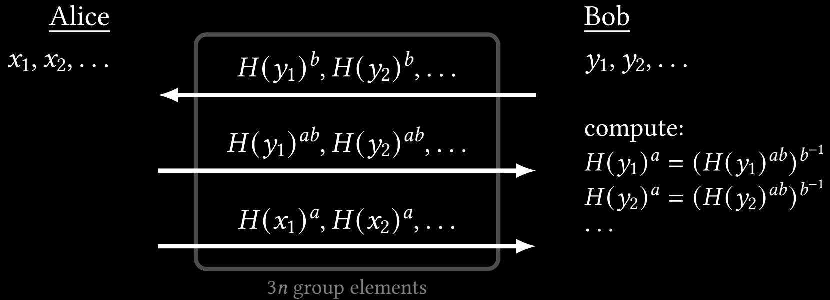
$H(y_1)^b, H(y_2)^b, \ldots$

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

$3n$ group elements

how could you possibly reduce communication?



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

$H(y_1)^{ab}, H(y_2)^{ab}, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

compute:

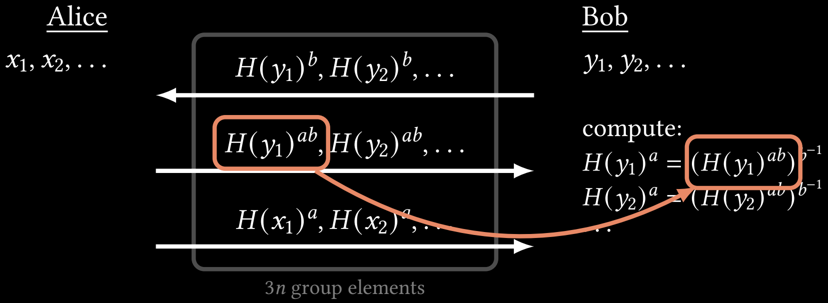
$H(y_1)^a = (H(y_1)^{ab})^{b^{-1}}$

$H(y_2)^a = (H(y_2)^{ab})^{b^{-1}}$

$\ldots$

$3n$ group elements

how could you possibly reduce communication?



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

$\boxed{H(y_1)^{ab}}, H(y_2)^{ab}, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

3n group elements

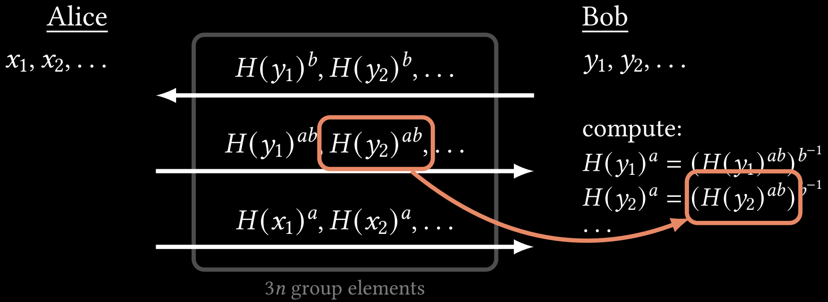
compute:

$H(y_1)^a = \boxed{(H(y_1)^{ab})}^{b^{-1}}$

$H(y_2)^a = (H(y_2)^{ab})^{b^{-1}}$

$\ldots$

how could you possibly reduce communication?



Alice

$x_1, x_2, \ldots$

$H(y_1)^b, H(y_2)^b, \ldots$

$H(y_1)^{ab}, \boxed{H(y_2)^{ab}}, \ldots$

$H(x_1)^a, H(x_2)^a, \ldots$

$3n$ group elements

Bob

$y_1, y_2, \ldots$

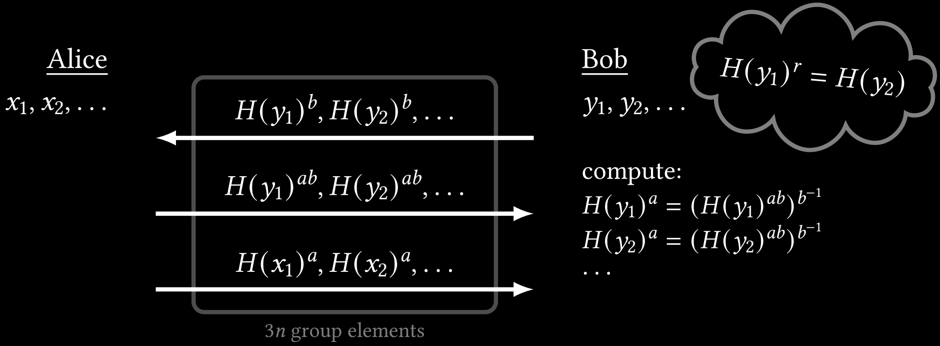
compute:

$H(y_1)^a = (H(y_1)^{ab})^{b^{-1}}$

$H(y_2)^a = \boxed{(H(y_2)^{ab})}^{b^{-1}}$

$\ldots$

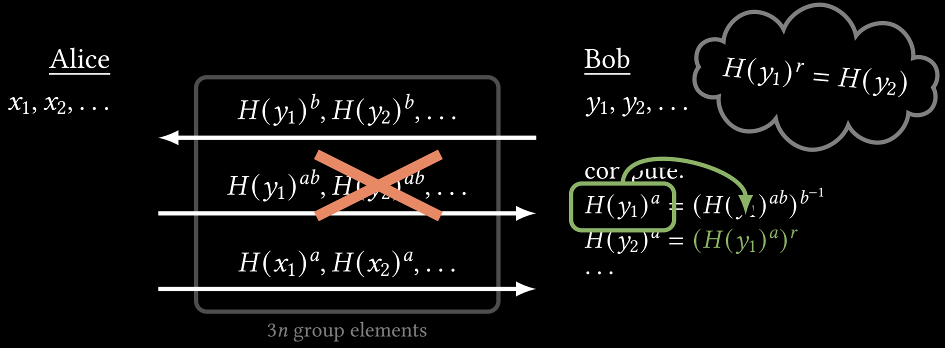
how could you possibly reduce communication?



how could you possibly reduce communication?

replace random oracle with some "trapdoored" function

… where Bob knows dlog relationships between outputs



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

$H(y_1)^r = H(y_2)$

$H(y_1)^b, H(y_2)^b, \ldots$

$\xcancel{H(y_1)^{ab}, H(y_2)^{ab}, \ldots}$

$H(x_1)^a, H(x_2)^a, \ldots$

3*n* group elements

compute:

$\boxed{H(y_1)^a} = (H(y_1)^{ab})^{b^{-1}}$

$H(y_2)^a = (H(y_1)^a)^r$

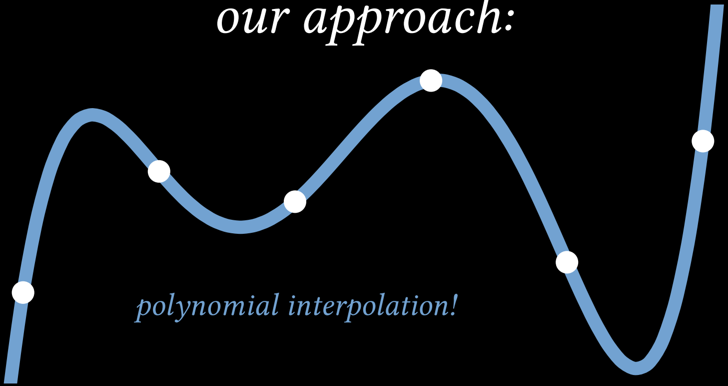
$\ldots$

how could you possibly reduce communication?

replace random oracle with some "trapdoored" function

… where Bob knows dlog relationships between outputs

our approach:



polynomial interpolation!

Alice	Bob
$x_1, x_2, \ldots$	$y_1, y_2, \ldots$

Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:

$P(y_i) = g^{b_i}$

Alice

$x_1, x_2, \ldots$

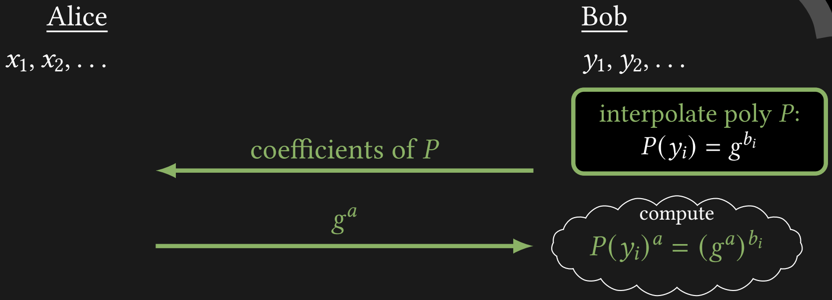
Bob

$y_1, y_2, \ldots$

interpolate poly P:

$P(y_i) = g^{b_i}$

coefficients of P



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:

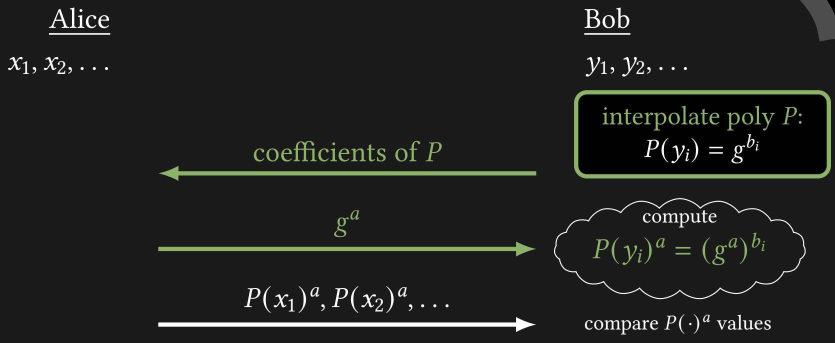
$P(y_i) = g^{b_i}$

coefficients of P

g^a

compute

$P(y_i)^a = (g^a)^{b_i}$



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = g^{b_i}$

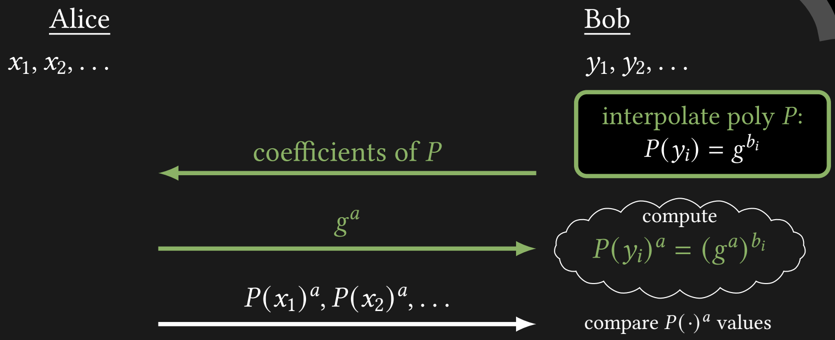
coefficients of P

g^a

compute
$P(y_i)^a = (g^a)^{b_i}$

$P(x_1)^a, P(x_2)^a, \ldots$

compare $P(\cdot)^a$ values



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = g^{b_i}$

coefficients of P

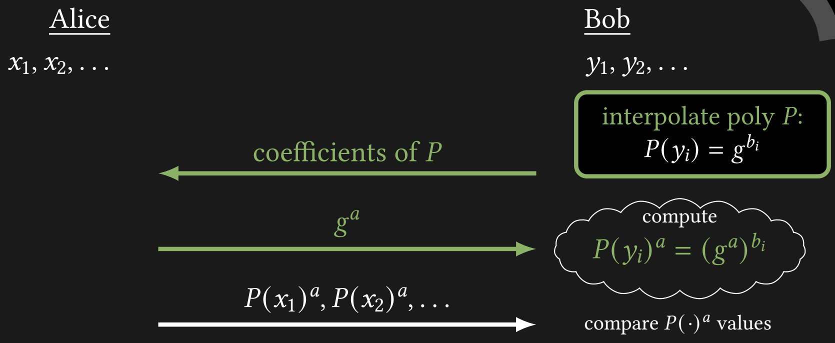
g^a

compute
$P(y_i)^a = (g^a)^{b_i}$

$P(x_1)^a, P(x_2)^a, \ldots$

compare $P(\cdot)^a$ values

correctness: Bob knows dlog of $P(y)$ for programmed points ✓



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = g^{b_i}$

coefficients of P ←

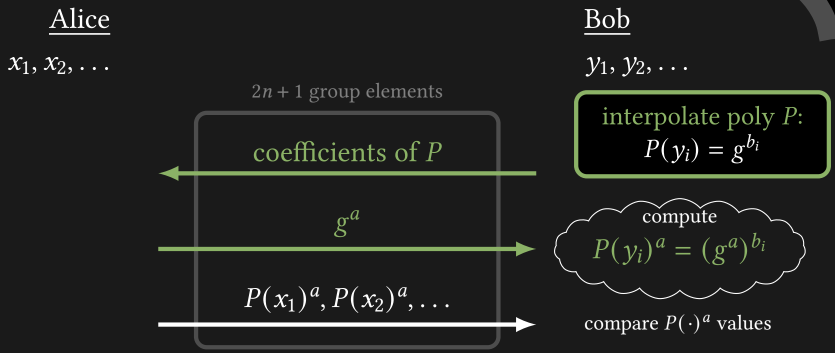
g^a →

compute
$P(y_i)^a = (g^a)^{b_i}$

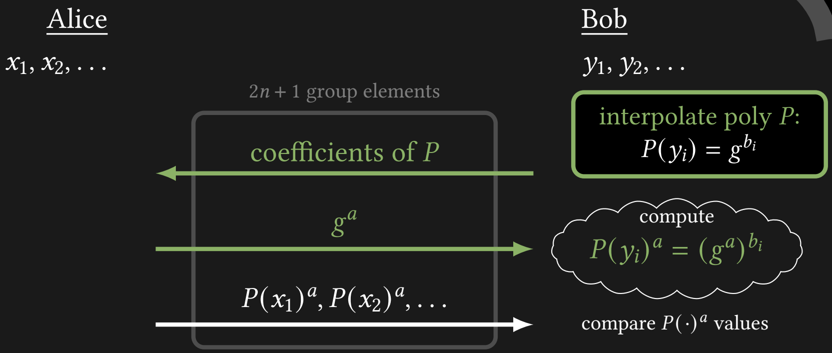
$P(x_1)^a, P(x_2)^a, \ldots$ →

compare $P(\cdot)^a$ values

correctness:	Bob knows dlog of $P(y)$ for programmed points ✓
obliviousness:	description of P doesn't leak choice of programmed points ✓



correctness:	Bob knows dlog of $P(y)$ for programmed points ✓
obliviousness:	description of P doesn't leak choice of programmed points ✓
efficiency:	$\lvert$description of $P\rvert = n$ group elements ✓



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = g^{b_i}$

$2n + 1$ group elements

coefficients of P ←

g^a →

$P(x_1)^a, P(x_2)^a, \ldots$ →

compute
$P(y_i)^a = (g^a)^{b_i}$

compare $P(\cdot)^a$ values

correctness: Bob knows dlog of $P(y)$ for programmed points ✓

obliviousness: description of P doesn't leak choice of programmed points ✓

efficiency: $|\text{description of } P| = n$ group elements ✓

$P(\cdot)^a$ is PRF: Bob **cannot know** dlog of any *other* $P(x)$??

interpolate so that:

$$P(y_i) = g^{b_i}$$

$$??\Downarrow??$$

other $P(x)$ outputs
have unknown dlog

interpolate so that:

$P(y_i) = g^{b_i}$

$? ? \Downarrow ? ?$

other $P(x)$ outputs
have unknown dlog

Ideal permutation model: all parties have oracle access to random Π, Π^{-1}

interpolate so that:
$P(y_i) = g^{b_i}$

? ? $\Downarrow$? ?

other $P(x)$ outputs
have unknown dlog

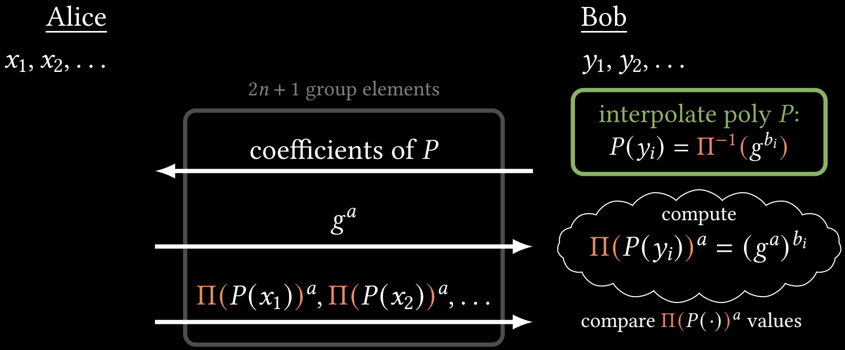
interpolate so that:
$P(y_i) = \Pi^{-1}(g^{b_i})$

$\Downarrow$ ✓

simulator can **program**
other $\Pi(P(x))$ outputs

Ideal permutation model: all parties have oracle access to random Π, Π^{-1}

our real protocol:



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:

$P(y_i) = \Pi^{-1}(g^{b_i})$

$2n + 1$ group elements

coefficients of P

g^a

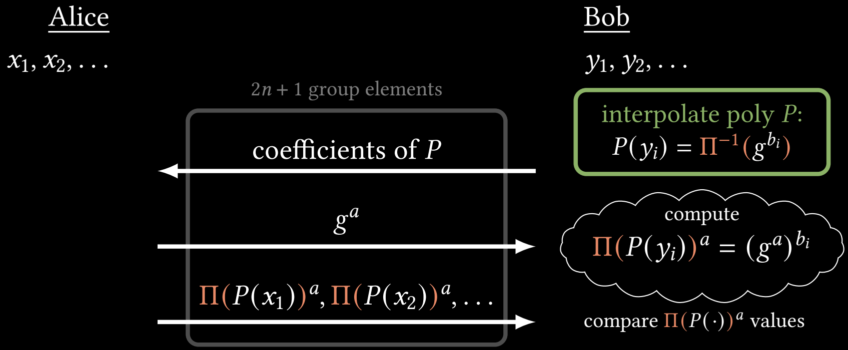
$\Pi(P(x_1))^a, \Pi(P(x_2))^a, \ldots$

compute

$\Pi(P(y_i))^a = (g^a)^{b_i}$

compare $\Pi(P(\cdot))^a$ values

our real protocol (fine print):



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:

$P(y_i) = \Pi^{-1}(g^{b_i})$

$2n + 1$ group elements

coefficients of P ←

g^a →

$\Pi(P(x_1))^a, \Pi(P(x_2))^a, \ldots$ →

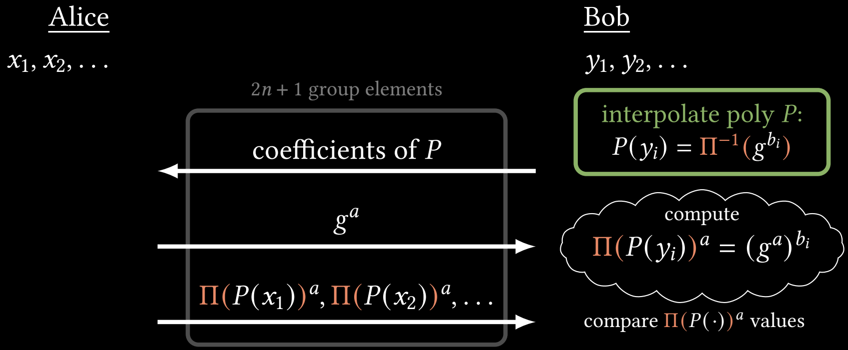
compute

$\Pi(P(y_i))^a = (g^a)^{b_i}$

compare $\Pi(P(\cdot))^a$ values

semi-honest: Alice's group elements can be truncated

our real protocol (fine print):



Alice

$x_1, x_2, \ldots$

Bob

$y_1, y_2, \ldots$

interpolate poly P:
$P(y_i) = \Pi^{-1}(g^{b_i})$

$2n + 1$ group elements

coefficients of P

g^a

$\Pi(P(x_1))^a, \Pi(P(x_2))^a, \ldots$

compute
$\Pi(P(y_i))^a = (g^a)^{b_i}$

compare $\Pi(P(\cdot))^a$ values

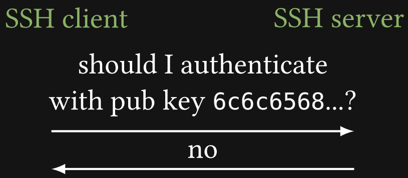
semi-honest: Alice's group elements can be truncated

malicious: a few more strategic RO calls (to help simulator extract)

Practical Privacy-Preserving Authentication for SSH

Lawrence Roy
Stanislav Lyakhov
Yeongjin Jang
Mike Rosulek

appeared at USENIX Security 2022

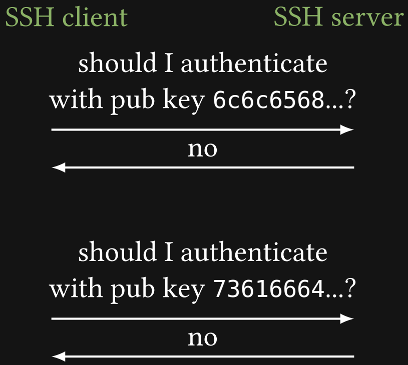


SSH client

SSH server

should I authenticate
with pub key `6c6c6568`…?

no



SSH client

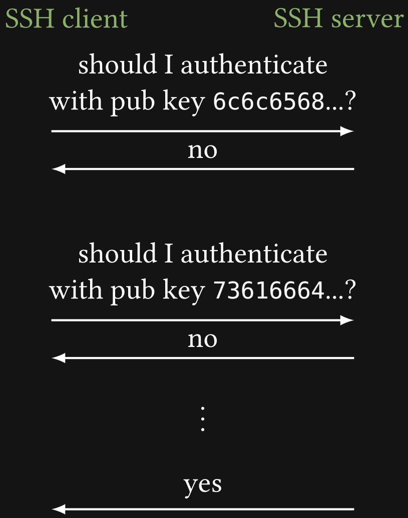
SSH server

should I authenticate
with pub key `6c6c6568`...?

no

should I authenticate
with pub key `73616664`...?

no



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`...?

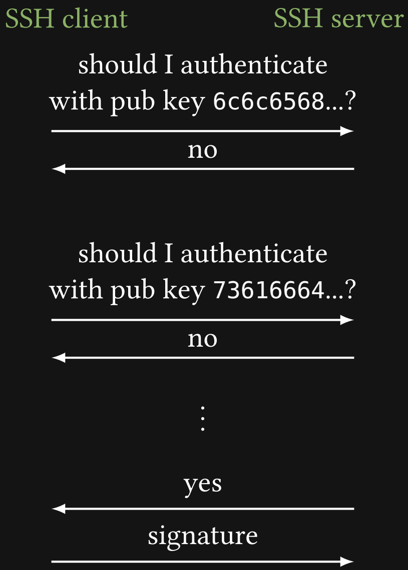
no

should I authenticate
with pub key `73616664`...?

no

⋮

yes



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`...?

no

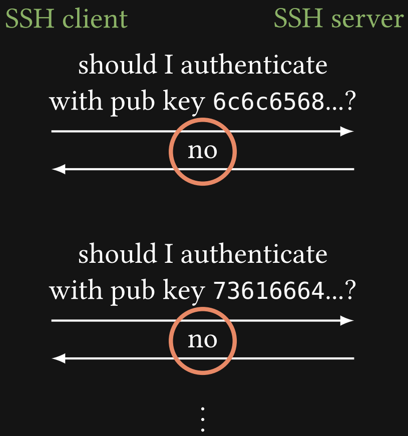
should I authenticate
with pub key `73616664`...?

no

⋮

yes

signature



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`...?

no

should I authenticate
with pub key `73616664`...?

no

⋮

problem: server can fingerprint client:

- refuse all advertisements ⇒ learn all keys

SSH client

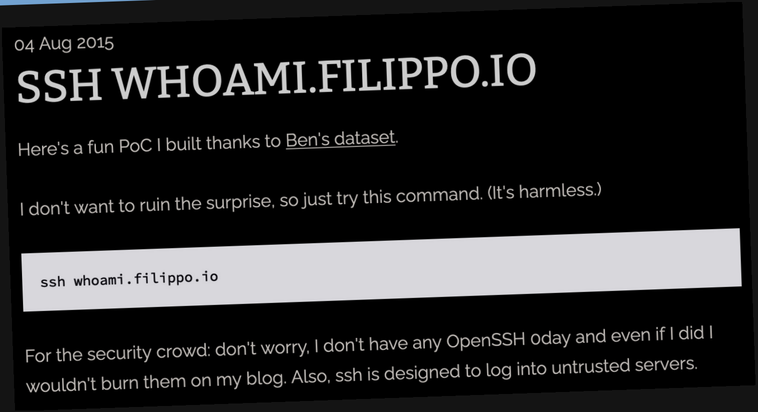
SSH server

problem: server can f

shou
with p

shou
with pu

l keys



04 Aug 2015

SSH WHOAMI.FILIPPO.IO

Here's a fun PoC I built thanks to Ben's dataset.

I don't want to ruin the surprise, so just try this command. (It's harmless.)

```
ssh whoami.filippo.io
```

For the security crowd: don't worry, I don't have any OpenSSH 0day and even if I did I wouldn't burn them on my blog. Also, ssh is designed to log into untrusted servers.

Filippo Valsorda `https://words.filippo.io/ssh-whoami-filippo-io/`

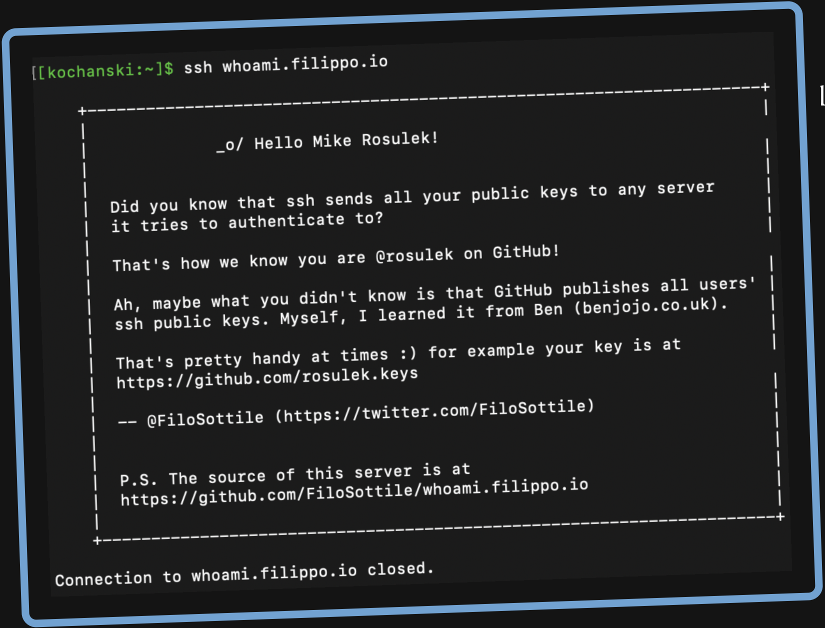
SSH client

shou

with p

shou

with pu



```
[[kochanski:~]$ ssh whoami.filippo.io

    +---------------------------------------------------------------------------+
    |                                                                           |
    |           _o/ Hello Mike Rosulek!                                         |
    |                                                                           |
    |                                                                           |
    |  Did you know that ssh sends all your public keys to any server           |
    |  it tries to authenticate to?                                             |
    |                                                                           |
    |  That's how we know you are @rosulek on GitHub!                           |
    |                                                                           |
    |  Ah, maybe what you didn't know is that GitHub publishes all users'       |
    |  ssh public keys. Myself, I learned it from Ben (benjojo.co.uk).          |
    |                                                                           |
    |  That's pretty handy at times :) for example your key is at               |
    |  https://github.com/rosulek.keys                                          |
    |                                                                           |
    |  -- @FiloSottile (https://twitter.com/FiloSottile)                        |
    |                                                                           |
    |                                                                           |
    |  P.S. The source of this server is at                                     |
    |  https://github.com/FiloSottile/whoami.filippo.io                         |
    |                                                                           |
    +---------------------------------------------------------------------------+

Connection to whoami.filippo.io closed.
```

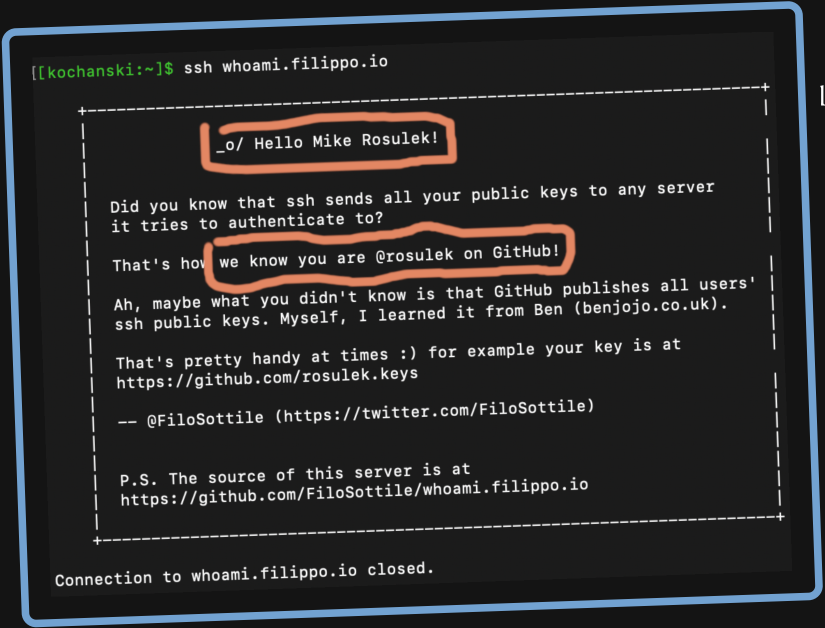
l keys

SSH client

shou
with p

shou
with pu

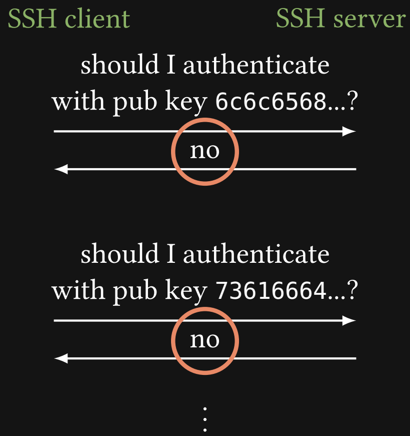
l keys



```
[[kochanski:~]$ ssh whoami.filippo.io

    +---------------------------------------------------------------------+
    |                                                                     |
    |              _o/ Hello Mike Rosulek!                                |
    |                                                                     |
    |                                                                     |
    |    Did you know that ssh sends all your public keys to any server   |
    |    it tries to authenticate to?                                     |
    |                                                                     |
    |    That's how we know you are @rosulek on GitHub!                   |
    |                                                                     |
    |    Ah, maybe what you didn't know is that GitHub publishes all users' |
    |    ssh public keys. Myself, I learned it from Ben (benjojo.co.uk).  |
    |                                                                     |
    |    That's pretty handy at times :) for example your key is at       |
    |    https://github.com/rosulek.keys                                  |
    |                                                                     |
    |    -- @FiloSottile (https://twitter.com/FiloSottile)                |
    |                                                                     |
    |                                                                     |
    |    P.S. The source of this server is at                             |
    |    https://github.com/FiloSottile/whoami.filippo.io                 |
    |                                                                     |
    +---------------------------------------------------------------------+

Connection to whoami.filippo.io closed.
```



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`...?

no

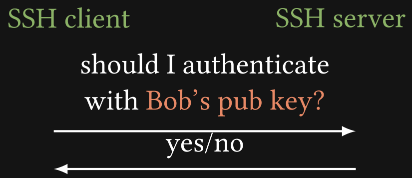
should I authenticate
with pub key `73616664`...?

no

⋮

problem: server can fingerprint client:

- refuse all advertisements ⇒ learn all keys
- can configure client to send only “correct” key

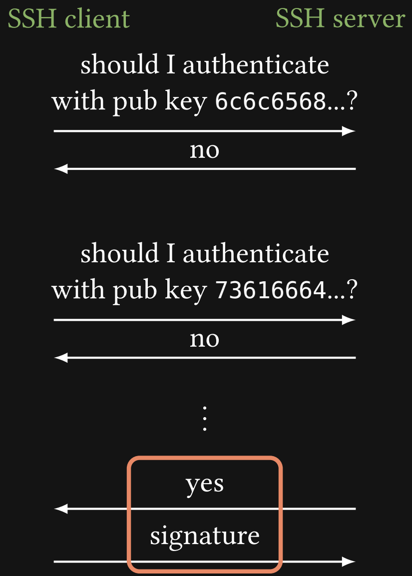


problem: server can fingerprint client:

- refuse all advertisements ⇒ learn all keys
- can configure client to send only “correct” key

problem: client can probe server:

- offer someone else’s pub key, observe response
- *pre-emptive* signatures possible (in principle)



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`…?

no

should I authenticate
with pub key `73616664`…?

no

⋮

yes

signature

problem: server can fingerprint client:

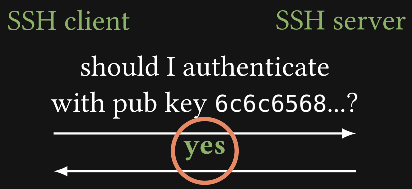
- refuse all advertisements ⇒ learn all keys
- can configure client to send only “correct” key

problem: client can probe server:

- offer someone else’s pub key, observe response
- *pre-emptive* signatures possible (in principle)

problem: server sees which key was used:

- and can **prove it!** ⇒ authentication not deniable
- fundamental to protocol



SSH client

SSH server

should I authenticate
with pub key `6c6c6568`…?

yes

problem: server can fingerprint client:

- refuse all advertisements ⇒ learn all keys
- can configure client to send only “correct” key

problem: client can probe server:

- offer someone else’s pub key, observe response
- *pre-emptive* signatures possible (in principle)

problem: server sees which key was used:

- and can **prove it!** ⇒ authentication not deniable
- fundamental to protocol

problem: server can act as honeypot:

- accept *any* key, even ones never seen before
- fundamental to protocol

goals of this work

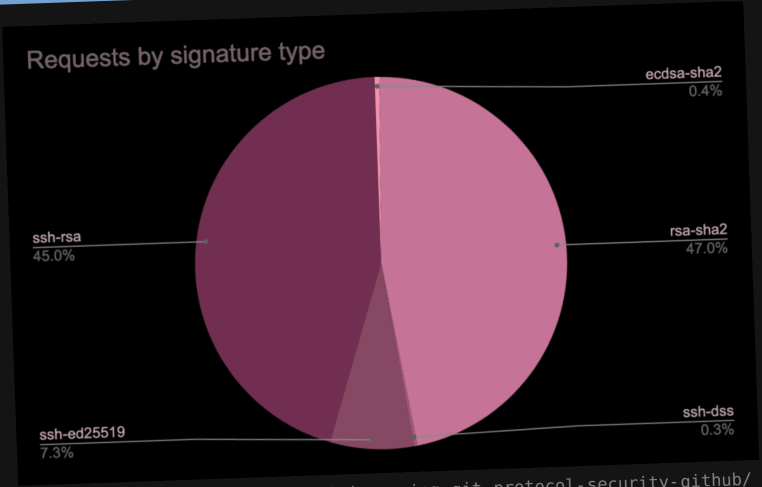
1 server & client should learn minimal information

goals of this work

1 server & client should learn minimal information

2 authenticate with respect to existing SSH keys

goals of this work

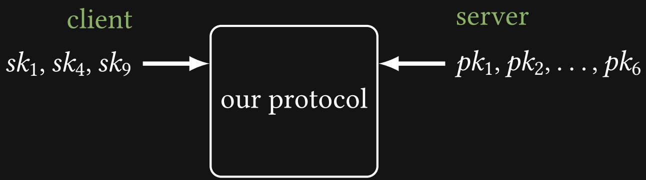


https://github.blog/2021-09-01-improving-git-protocol-security-github/

goals of this work

1. server & client should learn minimal information
2. authenticate with respect to existing SSH keys
3. minimize reliance on per-site configuration

our new authentication method: big picture



client

sk_1, sk_4, sk_9

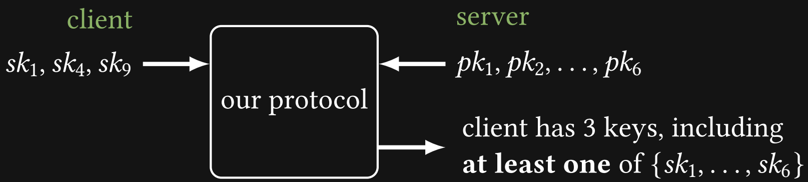
our protocol

server

$pk_1, pk_2, \ldots, pk_6$

- any **mixture** of existing RSA, ECDSA, EdDSA keys, in a single authentication attempt

our new authentication method: big picture



client

sk_1, sk_4, sk_9

our protocol

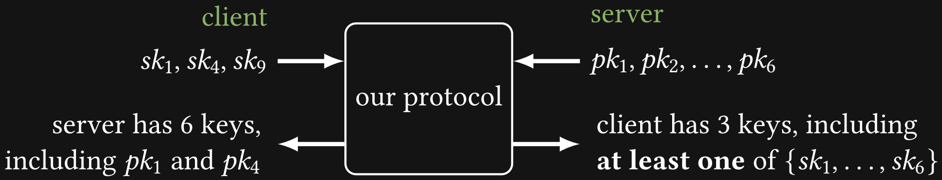
server

$pk_1, pk_2, \ldots, pk_6$

client has 3 keys, including **at least one** of $\{sk_1, \ldots, sk_6\}$

- any **mixture** of existing RSA, ECDSA, EdDSA keys, in a single authentication attempt

our new authentication method: big picture



client

sk_1, sk_4, sk_9

server

$pk_1, pk_2, \ldots, pk_6$

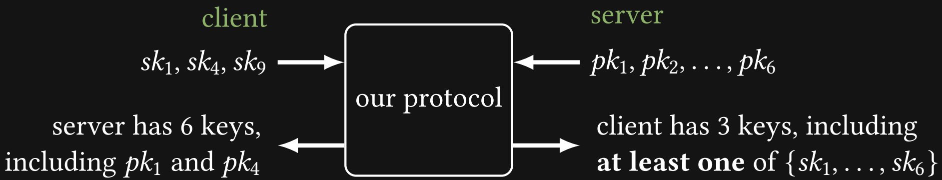
our protocol

server has 6 keys, including pk_1 and pk_4

client has 3 keys, including **at least one** of $\{sk_1, \ldots, sk_6\}$

- any **mixture** of existing RSA, ECDSA, EdDSA keys, in a single authentication attempt

our new authentication method: big picture



client

sk_1, sk_4, sk_9

server has 6 keys, including pk_1 and pk_4

our protocol

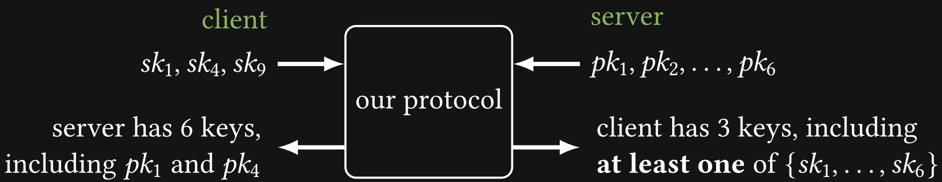
server

$pk_1, pk_2, \ldots, pk_6$

client has 3 keys, including **at least one** of $\{sk_1, \ldots, sk_6\}$

- any **mixture** of existing RSA, ECDSA, EdDSA keys, in a single authentication attempt
- does not depend on site-specific configuration; safe to use **all keys** in every authentication attempts

our new authentication method: big picture



client

sk_1, sk_4, sk_9

server

$pk_1, pk_2, \ldots, pk_6$

our protocol

server has 6 keys,
including pk_1 and pk_4

client has 3 keys, including
at least one of $\{sk_1, \ldots, sk_6\}$

- any **mixture** of existing RSA, ECDSA, EdDSA keys, in a single authentication attempt
- does not depend on site-specific configuration; safe to use **all keys** in every authentication attempts
- client won't connect unless server **knows** and **explicitly includes** one of client's keys

technical overview

client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

technical overview

client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

technical overview

client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$\xleftarrow{c}$

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

technical overview

client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$$c, \{m_j\}_j \leftarrow \mathsf{Enc}\Big(\{pk_j\}_j\Big)$$

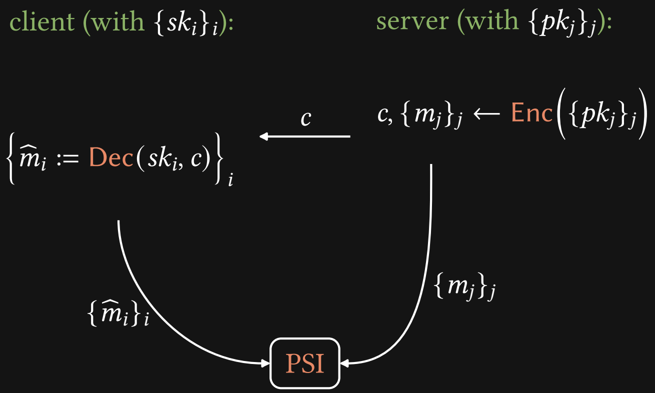
$$\xleftarrow{\;c\;}$$

$$\Big\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\Big\}_i$$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

technical overview



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$$c, \{m_j\}_j \leftarrow \mathsf{Enc}\big(\{pk_j\}_j\big)$$

c

$$\big\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\big\}_i$$

$\{\widehat{m}_i\}_i$

$\{m_j\}_j$

PSI

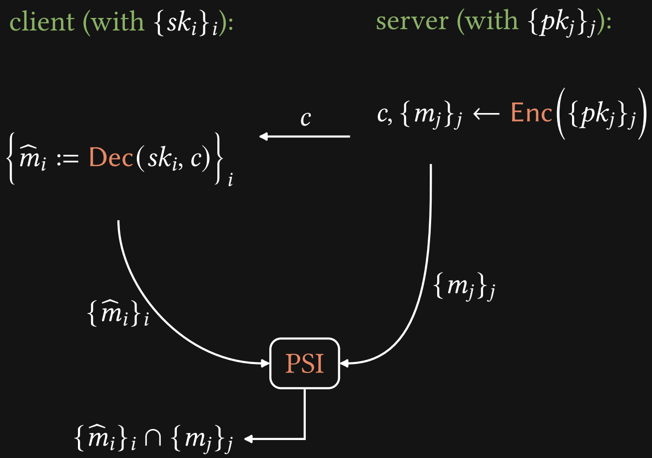
1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

2. private set intersection

each party has set of items;

technical overview



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

c

$\left\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\right\}_i$

$\{\widehat{m}_i\}_i$

$\{m_j\}_j$

PSI

$\{\widehat{m}_i\}_i \cap \{m_j\}_j$

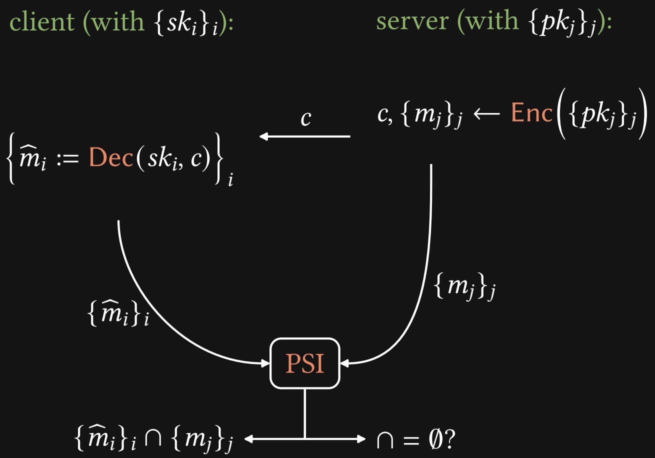
1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

2. private set intersection

each party has set of items;
client learns intersection;

technical overview



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

c

$\left\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\right\}_i$

$\{\widehat{m}_i\}_i$

$\{m_j\}_j$

PSI

$\{\widehat{m}_i\}_i \cap \{m_j\}_j$

$\cap = \emptyset?$

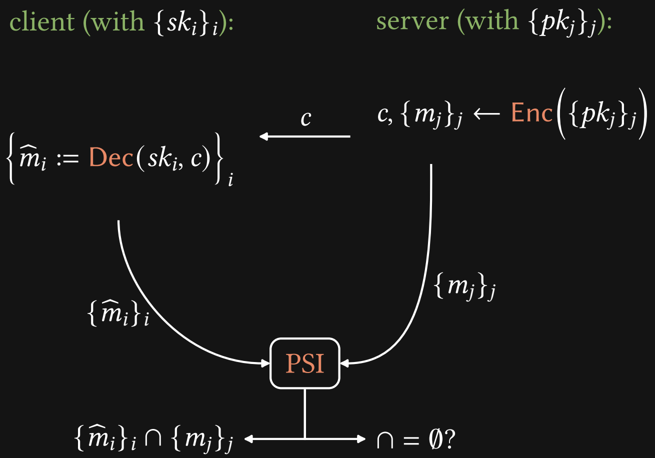
1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

2. private set intersection

each party has set of items;
client learns intersection;
server learns whether empty

technical overview & contributions



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

c

$\left\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\right\}_i$

$\{m_j\}_j$

$\{\widehat{m}_i\}_i$

PSI

$\{\widehat{m}_i\}_i \cap \{m_j\}_j$

$\cap = \emptyset?$

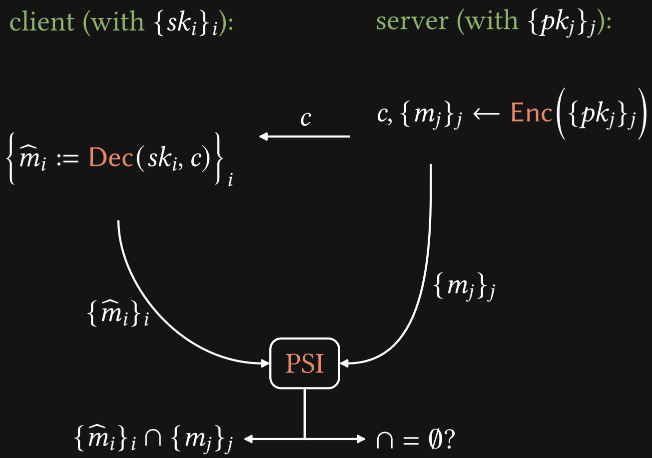
1. anonymous multi-KEM

single MKEM construction supporting RSA, ECDSA, & EdDSA

2. private set intersection

each party has set of items;
client learns intersection;
server learns whether empty

technical overview & contributions



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

c

$\left\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\right\}_i$

$\{\widehat{m}_i\}_i$

$\{m_j\}_j$

PSI

$\{\widehat{m}_i\}_i \cap \{m_j\}_j$

$\cap = \emptyset$?

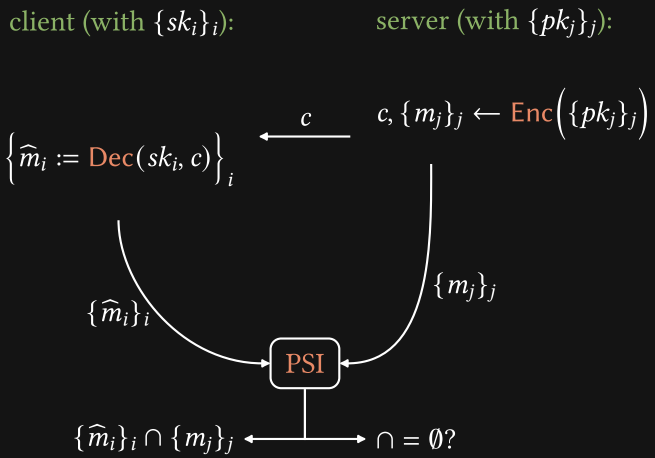
1. anonymous multi-KEM

single MKEM construction supporting RSA, ECDSA, & EdDSA

2. private set intersection

add "proof of nonempty intersection" to [RosulekTrieu21] PSI

technical overview & contributions



client (with $\{sk_i\}_i$):

server (with $\{pk_j\}_j$):

$c, \{m_j\}_j \leftarrow \mathsf{Enc}\left(\{pk_j\}_j\right)$

c

$\left\{\widehat{m}_i := \mathsf{Dec}(sk_i, c)\right\}_i$

$\{\widehat{m}_i\}_i$

$\{m_j\}_j$

PSI

$\{\widehat{m}_i\}_i \cap \{m_j\}_j$

$\cap = \emptyset?$

1. anonymous multi-KEM

single MKEM construction supporting RSA, ECDSA, & EdDSA

2. private set intersection

add “proof of nonempty intersection” to [RosulekTrieu21] PSI

+ full UC security analysis

concrete performance (in OpenSSH):

# of keys		RSA keys only (worst case for us)		{EC,Ed}DSA keys only (best case for us)	
client	server	time	comm	time	comm

`github.com/osu-crypto/PSIPK-ssh`

2 commodity desktop computers on LAN

concrete performance (in OpenSSH):

# of keys		RSA keys only (worst case for us)		{EC,Ed}DSA keys only (best case for us)	
client	server	time	comm	time	comm
5	10	60 ms	12 kB	9 ms	8 kB

`github.com/osu-crypto/PSIPK-ssh`

2 commodity desktop computers on LAN

concrete performance (in OpenSSH):

# of keys		RSA keys only (worst case for us)		{EC,Ed}DSA keys only (best case for us)	
client	server	time	comm	time	comm
5	10	60 ms	12 kB	9 ms	8 kB
20	100	320 ms	53 kB	28 ms	12 kB

`github.com/osu-crypto/PSIPK-ssh`

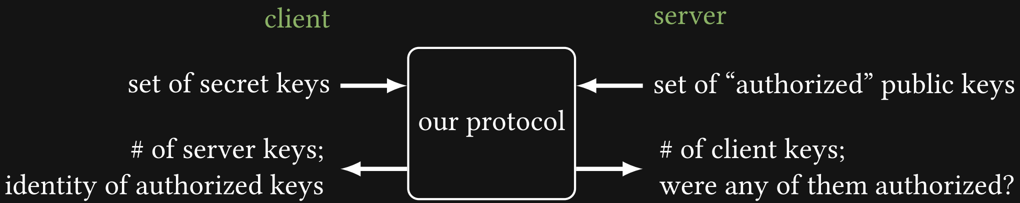
2 commodity desktop computers on LAN

concrete performance (in OpenSSH):

# of keys		RSA keys only (worst case for us)		{EC,Ed}DSA keys only (best case for us)	
client	server	time	comm	time	comm
5	10	60 ms	12 kB	9 ms	8 kB
20	100	320 ms	53 kB	28 ms	12 kB
20	1000	1200 ms	460 kB	214 ms	41 kB

`github.com/osu-crypto/PSIPK-ssh`

2 commodity desktop computers on LAN



client

server

set of secret keys

our protocol

set of “authorized” public keys

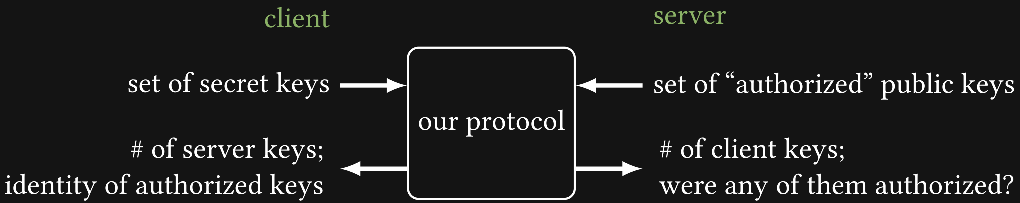
# of server keys;
identity of authorized keys

# of client keys;
were any of them authorized?

- ✓ efficient, practical
- ✓ mixture of existing RSA & EC keys
- ✓ safe without special per-site configuration

`github.com/osu-crypto/PSIPK-ssh`

`ia.cr/2022/740`



client

server

set of secret keys

our protocol

set of “authorized” public keys

\# of server keys;
identity of authorized keys

\# of client keys;
were any of them authorized?

- ✓ efficient, practical
- ✓ mixture of existing RSA & EC keys
- ✓ safe without special per-site configuration

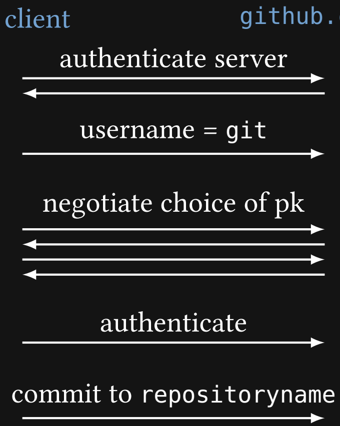
thanks!

github.com/osu-crypto/PSIPK-ssh

ia.cr/2022/740

(backup slides)

github over SSH:



client `github.com`

authenticate server

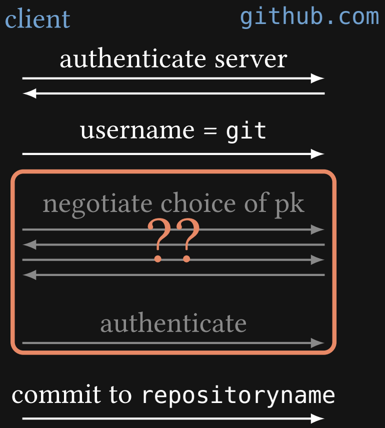
username = `git`

negotiate choice of pk

authenticate

commit to `repositoryname`

github over SSH:



client

`github.com`

authenticate server

username = `git`

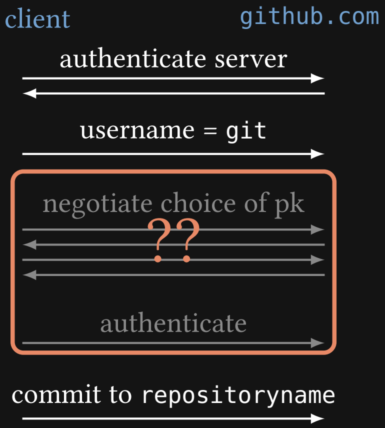
negotiate choice of pk

??

authenticate

commit to `repositoryname`

github over SSH:



client

`github.com`

authenticate server

username = `git`

negotiate choice of pk

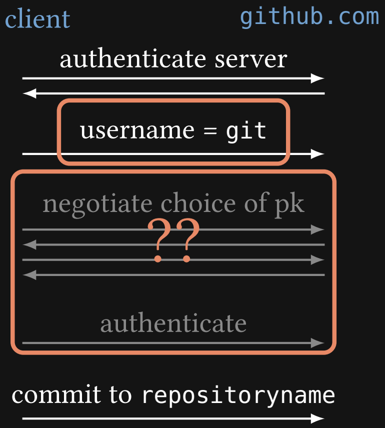
??

authenticate

commit to `repositoryname`

- server must decide **set of authorized keys** before running our protocol!

github over SSH:



client

`github.com`

authenticate server

username = `git`

negotiate choice of pk

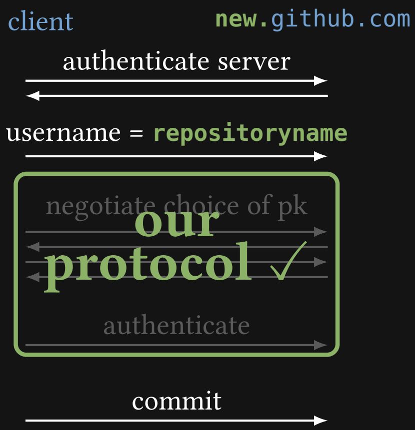
??

authenticate

commit to `repositoryname`

- server must decide **set of authorized keys** before running our protocol!
- server does not know repository name yet!

github over SSH:



- server must decide **set of authorized keys** before running our protocol!
- server does not know repository name yet!
- use repository name as `username`

anonymous multi-KEM

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of EdDSA/ECDSA

Alice: $pk_A = g^a$

Bob: $pk_B = g^b$

Charlie: $pk_C = g^c$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of EdDSA/ECDSA

Alice: $pk_A = g^a$

Bob: $pk_B = g^b$

Charlie: $pk_C = g^c$

$\xleftarrow{\text{ciphertext} = g^r}$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of EdDSA/ECDSA

Alice: $pk_A = g^a$

Bob: $pk_B = g^b$

Charlie: $pk_C = g^c$

ciphertext = g^r ←

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

Alice will decrypt to $(pk_A)^r$

Bob will decrypt to $(pk_B)^r$

Charlie will decrypt to $(pk_C)^r$

the case of EdDSA/ECDSA

Alice: $pk_A = g^a$

Bob: $pk_B = g^b$

Charlie: $pk_C = g^c$

ciphertext = g^r ←

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

Alice will decrypt to $(pk_A)^r$

Bob will decrypt to $(pk_B)^r$

Charlie will decrypt to $(pk_C)^r$

ciphertext hides set of recipients; even # of them!

the case of RSA

Alice: $pk_A = (N_A, e_A)$
Bob: $pk_B = (N_B, e_B)$
Charlie: $pk_C = (N_C, e_C)$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of RSA

Alice: $pk_A = (N_A, e_A)$

Bob: $pk_B = (N_B, e_B)$

Charlie: $pk_C = (N_C, e_C)$

encrypt $(r_A)^{e_A} \bmod N_A$

encrypt $(r_B)^{e_B} \bmod N_B$

encrypt $(r_C)^{e_C} \bmod N_C$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of RSA

Alice: $pk_A = (N_A, e_A)$

Bob: $pk_B = (N_B, e_B)$

Charlie: $pk_C = (N_C, e_C)$

encrypt $(r_A)^{e_A} \bmod N_A$

encrypt $(r_B)^{e_B} \bmod N_B$

encrypt $(r_C)^{e_C} \bmod N_C$

interpolate poly P:

$$P(N_A) = (r_A)^{e_A}$$
$$P(N_B) = (r_B)^{e_B}$$
$$P(N_C) = (r_C)^{e_C}$$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

the case of RSA

Alice: $pk_A = (N_A, e_A)$

Bob: $pk_B = (N_B, e_B)$

Charlie: $pk_C = (N_C, e_C)$

encrypt $(r_A)^{e_A} \bmod N_A$

encrypt $(r_B)^{e_B} \bmod N_B$

encrypt $(r_C)^{e_C} \bmod N_C$

interpolate poly P:

$$P(N_A) = (r_A)^{e_A}$$
$$P(N_B) = (r_B)^{e_B}$$
$$P(N_C) = (r_C)^{e_C}$$

1. anonymous multi-KEM

address ciphertext to $\{pk_j\}_j$;
sk_j decrypts c to m_j;
c hides pk_j recipients

ciphertext = P ←

PSI with proof of nonempty intersection

2. private set intersection

each party has set of items;
client learns intersection;
server learns whether empty

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]

Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$

oblivious PRF (OPRF) paradigm for PSI

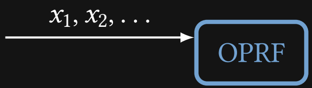
[FreedmanIshaiPinkasReingold05]

Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$



$x_1, x_2, \ldots$

OPRF

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]

Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$

$x_1, x_2, \ldots$

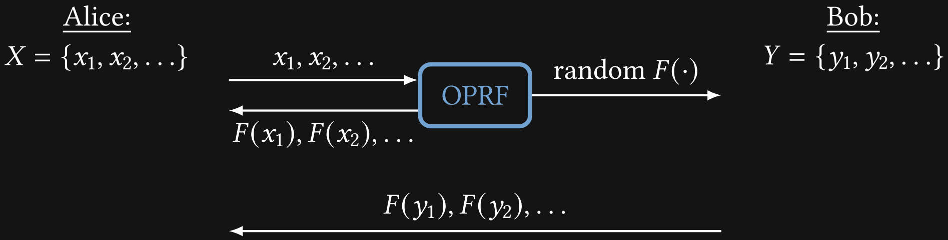
OPRF

random $F(\cdot)$

$F(x_1), F(x_2), \ldots$

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]



Alice:

$X = \{x_1, x_2, \ldots\}$

$x_1, x_2, \ldots$

OPRF

$F(x_1), F(x_2), \ldots$

random $F(\cdot)$

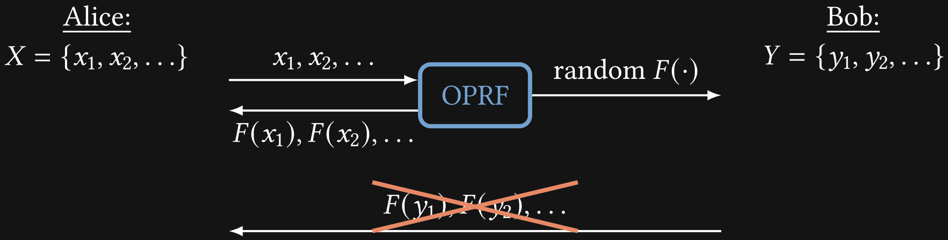
Bob:

$Y = \{y_1, y_2, \ldots\}$

$F(y_1), F(y_2), \ldots$

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]



Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$

$x_1, x_2, \ldots$

OPRF

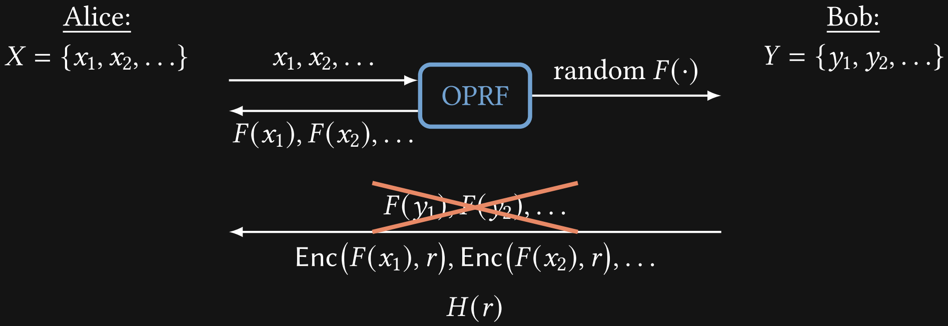
random $F(\cdot)$

$F(x_1), F(x_2), \ldots$

$\xcancel{F(y_1), F(y_2), \ldots}$

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]



Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$

$x_1, x_2, \ldots$

OPRF

random $F(\cdot)$

$F(x_1), F(x_2), \ldots$

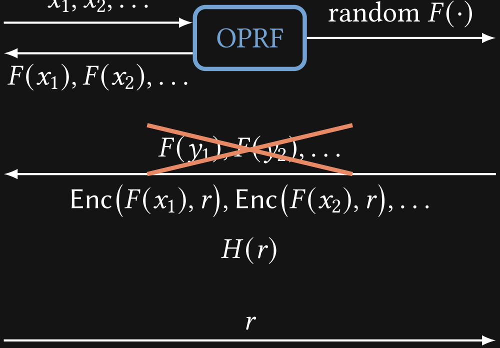
~~$F(y_1), F(y_2), \ldots$~~

$\mathsf{Enc}\big(F(x_1), r\big), \mathsf{Enc}\big(F(x_2), r\big), \ldots$

$H(r)$

oblivious PRF (OPRF) paradigm for PSI

[FreedmanIshaiPinkasReingold05]



Alice:

$X = \{x_1, x_2, \ldots\}$

Bob:

$Y = \{y_1, y_2, \ldots\}$

$x_1, x_2, \ldots$

OPRF

random $F(\cdot)$

$F(x_1), F(x_2), \ldots$

$\xcancel{F(y_1), F(y_2), \ldots}$

$\mathsf{Enc}\big(F(x_1), r\big), \mathsf{Enc}\big(F(x_2), r\big), \ldots$

$H(r)$

r