

Deploying Secure Computation Protocols in Daily Business Applications



Moti Yung

Google/ Columbia U.

**work done with the privacy
team @ **

Agenda

- **Background: Secure Computation: theory & practice (specific subject for general audience)**
 - **Meta level experience: Deployment vs. research in cryptography (general issue)**
 - **An actual experience: The Private Data Exchange (specific technical theme)**
 - **...Extensions....**
 - **Conclusions**
-

Secure Computing Protocols-- involved research area

- Started late 70's (as an outcome of Public Key Crypto just developed circa 76-77)...
 - Many interesting basic fundamental ideas (surprising & mathematically involved; viewed essentially as part of THEORY of Computing)
 - Great many researchers initiated its efforts! Many continue nowadays; very active area; nowadays experimentations, trials, initial attempts at actual systems are even taking place!
-

Secure Computation Protocols: “the Opening Lineup”

- [A. Shamir, R. Rivest, and L. Adleman](#), "Mental Poker", MIT TR 1979. (40+ years ago)
- [S. Goldwasser, and S. Micali](#). Probabilistic encryption & how to play mental poker keeping secret all partial information. STOC 82.
- [Michael O. Rabin](#). "How to exchange secrets by oblivious transfer." Harvard TR 81.
- [Manuel Blum](#), Coin Flipping by Telephone. CRYPTO'81
- [Andrew C. Yao](#). Protocols for Secure Computations, STOC 82.

RECREATIONAL PROBLEMS?? Well..... Look at this:

Secure Computation Protocols: the Opening Lineup

- A. Shamir, R. Rivest, and L. Adleman, "Mental Poker", MIT TR 1979. (Turing Award 2002)
 - S. Goldwasser, and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. STOC 82. (Turing Award 2012)
 - Michael O. Rabin. "How to exchange secrets by oblivious transfer." Harvard TR 81. (Turing Award 1976)
 - Manuel Blum, Coin Flipping by Telephone. CRYPTO'81 (Turing Award 1995)
 - Andrew C. Yao. Protocols for Secure Computations, STOC 82. (Turing Award 2000)
-

What is new in this set of applications?

- Traditional cryptography: a channel between two parties. Adversary is an outsider!!
- **Secure computing**: Adversary is controlling insiders. No need to assume external eavesdropper/ disruptor/ etc.
 - A talk to B and they are mutually distrusted
 - In some way this abstracts “privacy concerns:” (the adversary is internal to the system and mutual protection against insiders needed).

Development in general for the last ~40 years

- Theory: General Protocol many many results...
- General Secure Computing: Two party can compute any function without learning the other party's input (Yao 86)
- Multi Party computations: Compute any function with secret inputs, various settings, e.g., w/honest majority/ $\frac{2}{3}$ majority malicious failures (GMW 86, GHY87, BGW, CCD, R89...,)
- Modeling cryptographic functionalities, composability in protocols, adversary models (malicious, game theoretic,..).
- Partial Information Games (private inputs).

Development in general: specific protocols

- Theory of Special protocols: Many interesting results regarding specific protocols of high interest:
 - Election/ Voting protocols,
 - Payments (e-cash, cryptocurrencies: bitcoin),
 - Auctions, etc. (general computation results are typically inefficient)
 - Big Data → Privacy Preserving Communication/ Credentials (Chaum); Data Mining as an application (Lindell Pinkas).
-

Recent Positive Developments: Applied Sec. Comp.

- Practice: More Emphasize of communication and running time optimizations/ benchmarking (Usenix/ CCS/ S&P having works on implementation reports of optimized protocols!!!).
- :-) Some demos that distributed is useful: In use, simple comm. systems employing crypto, for some computations, special protocols:
 - Helios election, etc.;
 - Bitcoin (public repository/ agreement) 2009;
 - Threshold Cryptosystems: secure distributed RSA signing by CertCo in 97.....
 - Auctions based on secure computing...

Recent Positive Developments: Applied Sec. Comp. II

- :-) Initial MPC protocol for use: a protocol for auctions for Danish Farmers bidding (2009), the first showing multi-party approach is doable in dedicated application (share inputs computes on linear secret sharing).
- other apps: Estonia: tax checking (in progress).... Etc. etc.:-)
- A few startups in the area of secure computations (e.g. for key management, for ML, etc.)!!!

These are all dedicated applications, to show to business people that it may work.

BUT: what is the killer application in established business! :-)
(I am a cryptographer in Industry, working also on secure computing for >30 years.....)!!

What would be considered a business deployment of secure computing technology?

Survey of Cryptographers:

- One cryptographer: A business application which runs routinely!
 - Second cryptographer: High impact business application!
 - Third cryptographer: When I suggest such protocols, I am told no engineering team in the company will be able to implement them! So, I concluded it is good theory, too hard to spend time on commercializing this!
-

Core Business Deployment:

All I said, there is a lot of activity to build libraries/ demos/ etc.
and very specialized designs

Special applications are a good start. But, what about in an
established business?

- SINCE: Theory + experiments + demo: Together solve about 10% of the ``actual deployment puzzle'' of any reasonably complex problem in an established company!!
-

Business...., how to start?

- Need **incentives/ clear benefits**.
- The need for crypto may come from **different reasons**.
- Needed: Awareness/ knowledge of the business issues/ engineering/ product plans/ software development plan:
→ Need to play ``**Product manager**'' role
- Propose **solutions**: what actual problem it solves (and why it is uniquely positioned-- i.e. no alternatives).
- Needed: Where and how to use the opportunity in the **overall product context** (as enabler/ preventer)?
- → Start with the real **problems/ issues**! (Problem Solving)

REVISIT: Three Generation of Open Modern Cryptographic Technology

- 1. Symmetric Cryptography: DES 73 (standard 77):
Main driver **inter banking communications**
- 2. Public key cryptography [DH, RSA] 76, 77: Main
use **distributed systems, Internet SSL/TLS.**
- 3. Secure computation Protocols 78,79:..... **????**

Use of the first two generation

No alternative as communication in computer networks (Decnet, IBM's SNA,..Internet, Storage): banks are distributed, Internet, Mobile networks, Cloud Hosting, Infrastructures,..

..... for secure computing

Different situation....

alternatives.....



Third Generation: how to approach deployment?

- Till recently it was not considered needed in business.....
- No one even tried commercially.....BUT:
- When I joined Google I realized: Internet collaborative business + Cloud platforms/ hosting services + (now: mobile + IoT + big-data collaborations in analytics/ learning/...) → This is needed! Need first use! (Hence: also the startups!)
- Google is an engineering org; start with applications; build on it (rather than build on pure long research projects).

Initial Exploration- decide where to deploy:

Innovation as a Social/ tactical choice

- Offline Computation can tolerate computation delays, etc.
- Essential: critical to the company(!!!!)
- Involves data from different companies/ sources
- Have concrete privacy and sharing restrictions (user privacy regulations, trade secrets, etc.)
- Alternatives are all bad and will be rejected (by at least one of the parties: trust model insufficient: e.g., violation of regulations, etc.)

Secure Computing: In General

Why Now?

- (2012) Internet e-business: a multi company cooperation
- Cloud: data hosted outside (for users, mobile).
- Privacy is demanded in user data handling

All the above under increased privacy constraints! Alternatives less attractive!

Private Data Exchange

Concrete system System

- Content Provider **G**: Viewing users list
- Transaction Provider (Merchant) **M**: Spend values by users list [i.e., who paid how much]
- Goal:
 - Discover spend value for a set of users of G per merchant, to assess the correlation of viewing vs. spending (**compute: number of spenders; total spend value, [and leak upper bound on user lists' sizes]**).
 - **Raw Data can't be exchanged** → **Private Data Exchange**

Goal:

Find under constraints:

- (1) sum of spend values for common ids.
- (2) #common ids, (allow learning “some side info” like upper bounds on the sizes of lists to boost performance, as below:).

While:

(& 3) Performance wise: Minimizing Communication (**big data**)!! And performing reasonable computation (avoid excessive processing)

Goal: discovering spend- no privacy

G

G is set of Ids $\{G_1, G_2, \dots\}$
of viewers

M

M is Ids $\{M_1, M_2, \dots\}$ at merchant.

S is set of spend values. Merchant
holds: $(M_1, S_1), (M_2, S_2),$
...

1. $(M_1, S_1), (M_2, S_2), \dots$



2. Sum the spends for all Ids in
common.

- **Problem (!!!!!!!):**

- reveals to G all users
- reveals to G all users and their spending.

Goal:

Constrained Learning/ Output

Achieving PRIVACY under Cryptographic hiding:

- M finds **NOTHING** about:
 - ids of G
 - sum of spend values by viewing users
- G finds **ONLY AGGREGATE INFO**:
 - **size** of set of users in common
 - **total spent** values.

Privacy: Each side's privacy merely based on its own actions and practices for the duration of the protocol

Toward soln: Only protect spend values

E: homomorphic encryption (paillier) of M. $E(S_1) \times E(S_2) \rightarrow E(S_1 + S_2)$

G

M

1. $(M_1, E(S_1)), (M_2, E(S_2)), \dots$

2. Find common ids and their encrypted spends $E(S_i) \dots E(S_j)$

3. $E = E(S_i) \times E(S_{i+1}) \dots E(S_j)$

4. $S = S_i + S_{i+1} + S_j$ after homomorphic decryption

*** **PROBLEM: Reveals total spend to M**

Pailler 1999:

Security is based on factoring (the composite residuosity assumption) known to be broken only by factoring (like the RSA function).


2nd step: ...also protect total spend

Use Blinding: Blind Sum Protocol-- homomorphic encr.
under merchant key

G

M (owns E)

1. $(M_1, E(S_1)), (M_2, E(S_2)), \dots$



2. Find common ids and their encrypted
spends $E(S_i) \dots E(S_j)$

3. $E = E(R) \times E(S_i) \times E(S_{i+1}) \dots \times E(S_j)$



4. $S = R + S_i + S_{i+1} + S_j$ after homomorphic decryption



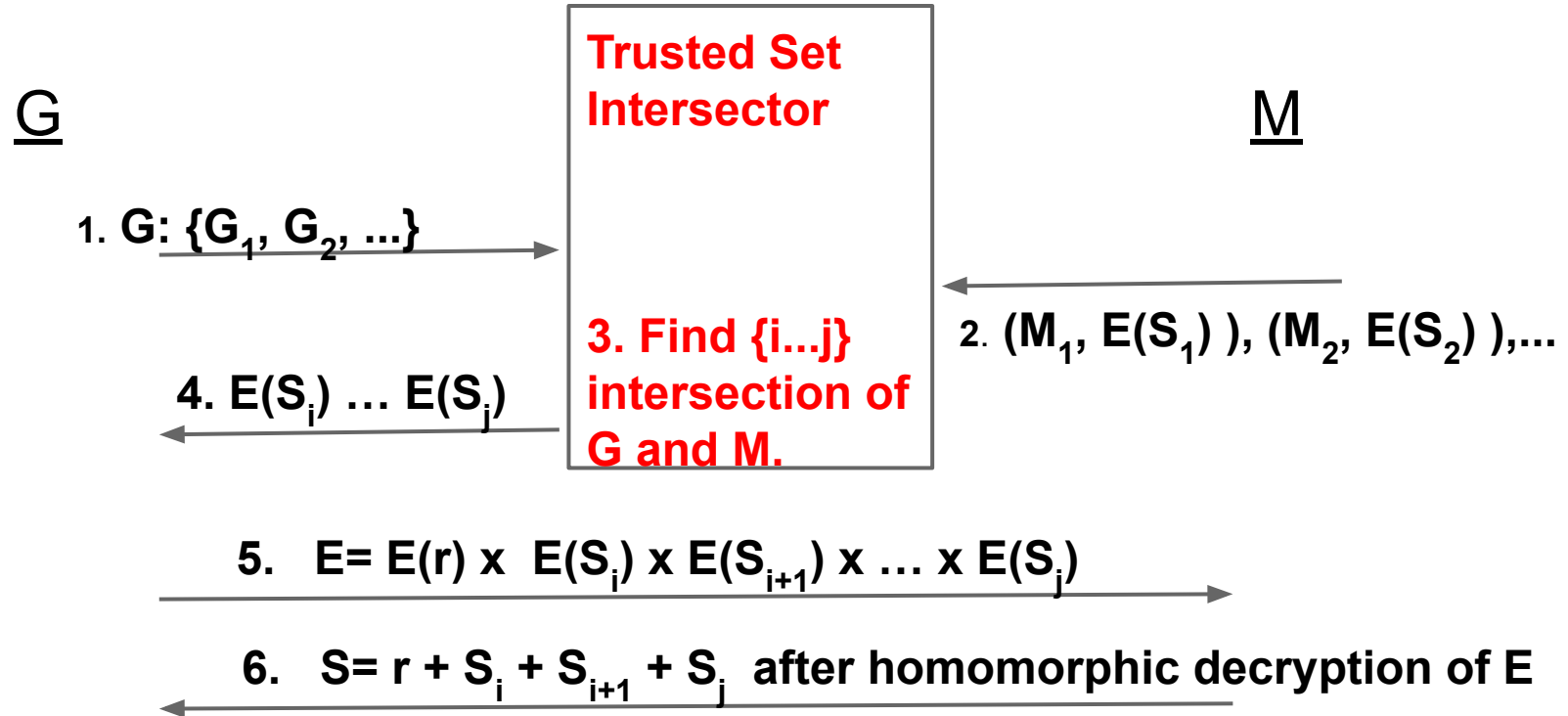
5. subtract R to get total spend (and nothing more!)

Problem: reveals M's IDs

- So far reveals all M's IDs to G
 - too much leakage....
 - also reveals that users spent (even if not the amount).
- Can we avoid revealing the IDs?
- Yes → (blinded) **private set intersection (PSI)**.

Note: PSI (very current and well studied problem) is a tool for the **private data exchange..**

Abstractly: Trusted set intersector



Removing trusted 3rd party

- ~~Trusted 3rd party~~ → based on privacy preserving blinded set intersection (PSI)..
- Numerous methods and extensive research to get PSI
- Use commutative block encryption hiding ID's:
 - **f, g: commutative encryption**
 - **$f(g(m))$ equals $g(f(m))$.**
- Combine with Blind Sum.

commutative encryption

- [Pohlig Hellman 76]

symmetric encryption $f(m) = m^{e_1} \bmod p$.

- $g(m) = m^{e_2} \bmod p$. [exponentiation cipher]
- $fg(m)$ is $m^{e_1 e_2} \bmod p$ and $gf(m) = m^{e_2 e_1} = m^{e_1 e_2} \bmod p$
 - Order of exponentiation not important! [Shamir 80]
 - Can do over elliptic curve groups too (smaller)
 - Can be viewed as “commutative joint hash”



Pohlig Hellman Security

Over DDH groups (late 90's idea) if we have random r and its encryption: and then, given:

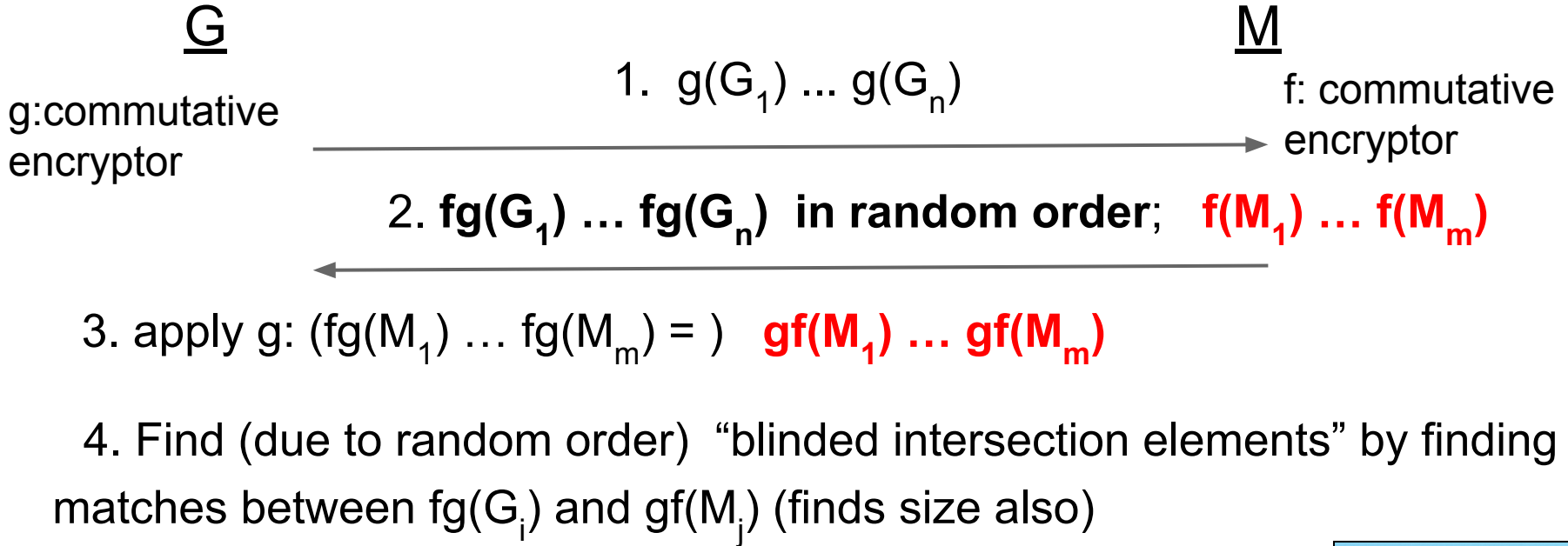
(1) encryption of random m or (2) random z ; we cannot tell

$$\langle r, r^{e_1}, m, m^{e_1} \rangle$$

$$\langle r, r^{e_1}, m, z \rangle$$

We will ROM-hash the ID's: $ID \rightarrow \text{SHA256}(ID)$, to get a “random generator m ” in the DDH assumption, the transformation is a PR family (per each exponent)...

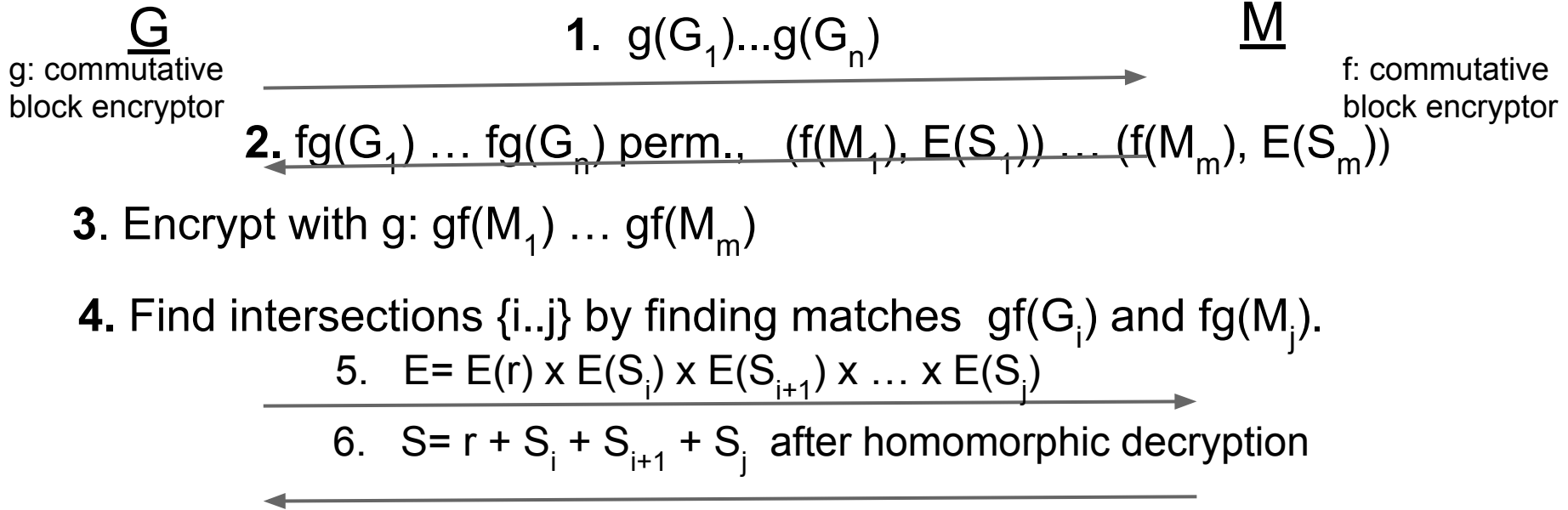
Privacy preserving set intersection



2-birds: if step 2 returned in order → set intersection



Complete solution: Int Cardinality-Sum



5 & 6: Blinded Sum based on Paillier (additive homomorph.)

Basic Solutions (honest parties)

Talking with eng. and clients: ...more to solve...

(1) Basic one: “common subset affine function”

(2) Reverse (requirement: summing at merchant who will allow to continue or not):

- each spending individually blinded (kept w/ encrypted ID)
- Merchants aggregate; G sends the relevant unblindings

Implication: first solution may not be enough- merchant side legal constraints have implications (--> talk with engineers, business people, clients).

-encrypt squares of spending w/Paillier can compute Standard Deviation as well....

Adding Solutions:

- Robustness: against malicious behavior

In practice: malicious/benign q. is based on trust.

- New tools to do it; solve related problems...

- Other methods (pseudorandom functions, OT, ...very interesting [may need more convincing]..)

- Optimizations (including leakage of side info vs. efficiency)

STATUS

- Implemented, in daily “big data” use
- First tries 2017. Open Sourced 2019
- Keys ephemeral, minimizing key storage req's
- Security “tested”: “cousins” of DH (Pohlig-H) and of RSA (Paillier) which secure the Internet!!
- Routine usage for these critical important cases of data analysis, keeping all PI and PII private. Adaptation to solve other issues



Publications 2020:

-- Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, Moti Yung:

On Deploying Secure Computing Commercially: Private Intersection-Sum Protocols and their Business Applications.

ePrint 2019: 723 (2019) [Euro S&P 2020](#)

-- Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, Moti Yung:

Two-Sided Malicious Security for Private Intersection-Sum with Cardinality. ePrint 2020: 385 (2020) [Crypto 2020](#)

OPEN SOURCED +BLOG: Private Join and Compute

Google Security Blog

The latest news and insights from Google on security and safety on the Internet

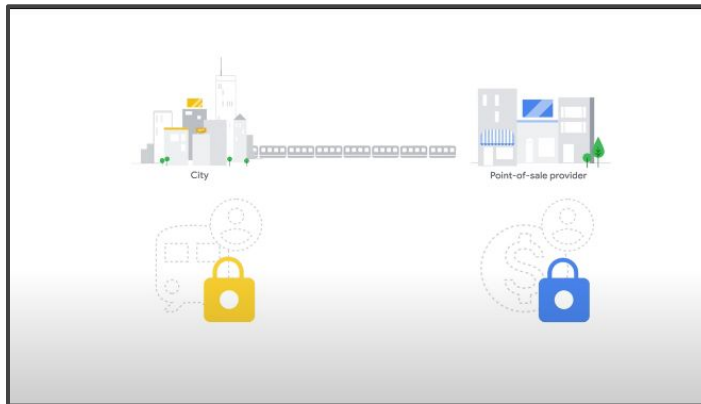
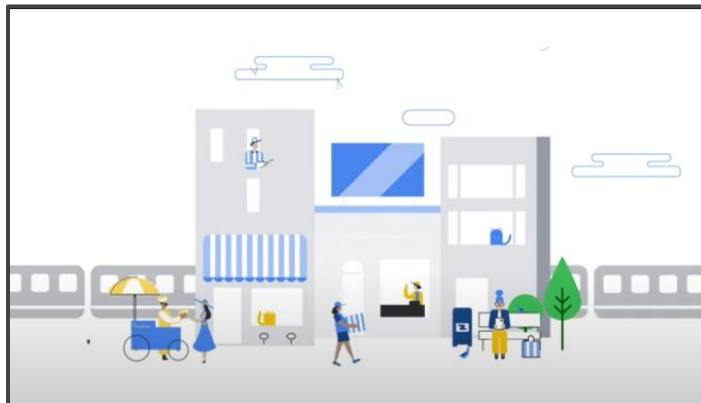
Helping organizations do more without collecting more data

June 19, 2019

Posted by Amanda Walker, Engineering Director; Sarvar Patel, Software Engineer; and Moti Yung, Research Scientist, Private Computing

We continually [invest in new research](#) to advance innovations that preserve individual privacy while enabling valuable insights from data. Earlier this year, we launched [Password Checkup](#), a Chrome extension that helps users detect if a username and password they enter on a website has been compromised. It relies on a cryptographic protocol known as [private set intersection \(PSI\)](#) to match your login's credentials against an encrypted database of over 4 billion credentials Google knows to be unsafe. At the same time, it ensures that no one – including Google – ever learns your actual credentials.

Today, we're rolling out the [open-source availability](#) of Private Join and Compute, a new type of [secure multi-party computation \(MPC\)](#) that augments the core PSI protocol to help organizations work together with confidential data sets while raising the bar for privacy.



Facebook Private Match + PS3I

facebook Engineering

Open Source ▾ Platforms ▾ Infrastructure Systems ▾ Physical Infrastructure ▾ Video Engineering & AR/VR ▾

POSTED ON JUL 10, 2020 TO DEVELOPER TOOLS, OPEN SOURCE

Private matching for compute: New solutions to the problem of enabling compute on private set intersections

Alice		
lanastasiades@example.com		\$50
annelopez82@example.net		\$100
williamfulmore@example.net		\$80
sebastian.reilly@example.net		\$25
cpaynter@example.com		\$60
jean.magee@example.com		
lottispillman@example.net		\$150
carljohnson44@example.com		\$30
cindyweiners@example.com		\$45

Bob		
annelopez82@example.net	Food	
richardwilliams@example.com	Community	
sebastian.reilly@example.net	Community	
dennis.jones@example.net	Food	
jaredmason@example.com	Community	
carljohnson44@example.com	Food	
cindyweiners@example.com	Community	

By Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, Vlad Vlasin

Matching records is one of the most basic data analysis operations. There are many cases where data needs to be aligned across some common value — whether that's joining between two different tables in a database or across two data sets stored in a file, or matching data sets between two separate entities.

Beyond private set intersection

The previous example performed a very simple analysis on a joined data set: a count of distinct items. It is often the case that we want to do something more complex, but we still want to rely on data from both Alice and Bob. More sophisticated PETs like multiparty computation (MPC) and homomorphic encryption (HE) allow Alice and Bob to perform various downstream computations on larger data sets while keeping everything except the final outcome of the computation protected. Consider the following examples:

1. Calculate the total money donated to different categories, where Alice knows the donation amounts and Bob knows people's category preferences.
2. Calculate the test statistics of a randomized control trial, comparing an outcome where the test outcomes are known to Alice and the treatment and control group memberships are known to Bob.
3. Train a machine learning (ML) model that estimates lifetime donations, where historical donations are known by Alice and the predictive features (e.g., years active and total events volunteered) are known by Bob.

The Malicious Adversary Work (just sketch of it):

- ~~Honest-but-curious-security~~ Security against Malicious/Active adversaries
- Both sides should receive the output (!!!!) [single side protocols exist]
- Communication cost + Monetary cost are more important than end-to-end runtime
- Communication cost 4-5x greater than semi-honest protocol based on DDH
- Monetary cost ~25x greater than semi-honest protocol based on DDH

There are Efficient one-sided Malicious-secure PSI

Efficient Set Intersection with Simulation-Based Security

Michael J. Freedman* Carmit Hazay† Kobbi Nissim‡ Benny Pinkas§

September 4, 2014

Abstract

We consider the problem of computing the intersection of private datasets of two parties, where the datasets contain lists of elements taken from a large domain. This problem has many applications for online collaboration. In this work we present protocols based on the use of homomorphic encryption and different hashing schemes for both the semi-honest and malicious environments. The protocol for the semi-honest environment is secure in the standard model, while the protocol for the malicious environment is secure in the random oracle model. Our protocols obtain linear communication and computation overhead. We further implement different variants of our semi-honest protocol. Our experiments show that the asymptotic overhead of the protocol is affected by different constants. (In particular, the degree of the polynomials evaluated by the protocol matters less than the number of polynomials that are evaluated.) As a result, the protocol variant with the best asymptotic overhead is not necessarily preferable for inputs of reasonable size.

Improved Private Set Intersection against Malicious Adversaries

Peter Rindal* Mike Rosulek*

October 3, 2016

Abstract

Private set intersection (PSI) refers to a special case of secure two-party computation in which the parties each have a set of items and compute the intersection of these sets without revealing any additional information. In this paper we present improvements to practical PSI providing security in the presence of malicious adversaries.

Our starting point is the protocol of Dong, Chen & Wen (CCS 2013) that is based on Bloom filters. We identify a bug in their malicious-secure variant and show how to fix it using a cut-and-choose approach that has low overhead while simultaneously avoiding one the main computational bottleneck in their original protocol. We also point out some subtleties that arise when using Bloom filters in malicious-secure cryptographic protocols.

We have implemented our PSI protocols and report on its performance. Our improvements reduce the cost of Dong et al.'s protocol by a factor of $14 - 110\times$ on a single thread. When compared to the previous fastest protocol of De Cristofaro et al., we improve the running time by $8 - 24\times$. For instance, our protocol has an online time of 14 seconds and an overall time of 2.1 minutes to securely compute the intersection of two sets of 1 million items each.

PSI from PaXoS: Fast, Malicious Private Set Intersection

Benny Pinkas* Mike Rosulek† Ni Trieu‡ Avishay Yanai‡

Abstract

We present a 2-party private set intersection (PSI) protocol which provides security against malicious participants, yet is almost as fast as the fastest known *semi-honest* PSI protocol of Kolesnikov et al. (CCS 2016).

Our protocol is based on a new approach for two-party PSI, which can be instantiated to provide security against either malicious or semi-honest adversaries. The protocol is unique in that the only difference between the semi-honest and malicious versions is an instantiation with different parameters for a linear error-correction code. It is also the first PSI protocol which is concretely efficient while having linear communication and security against malicious adversaries, while running in the OT-hybrid model (assuming a non-programmable random oracle).

State of the art semi-honest PSI protocols take advantage of cuckoo hashing, but it has proven a challenge to use cuckoo hashing for malicious security. Our protocol is the first to use cuckoo hashing for malicious-secure PSI. We do so via a new data structure, called a probe-and-XOR of strings (PaXoS), which may be of independent interest. This abstraction captures important properties of previous data structures, most notably garbled Bloom filters. While an encoding by a garbled Bloom filter is larger by a factor of $\Omega(\lambda)$ than the original data, we describe a significantly improved PaXoS based on cuckoo hashing that achieves constant rate while being no worse in other relevant efficiency measures.

Malicious-Secure Private Set Intersection via Dual Execution*

Peter Rindal† Mike Rosulek†

August 9, 2017

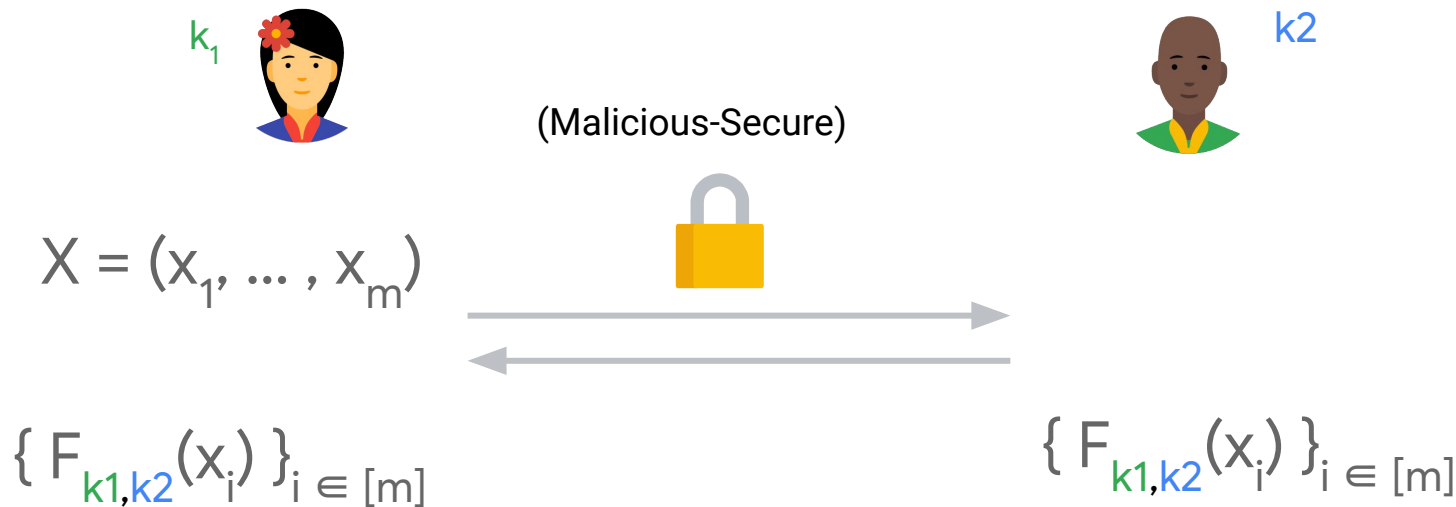
Abstract

Private set intersection (PSI) allows two parties, who each hold a set of items, to compute the intersection of those sets without revealing anything about other items. Recent advances in PSI have significantly improved its performance for the case of semi-honest security, making semi-honest PSI a practical alternative to insecure methods for computing intersections. However, the semi-honest security model is not always a good fit for real-world problems.

In this work we introduce a new PSI protocol that is secure in the presence of malicious adversaries. Our protocol is based entirely on fast symmetric-key primitives and inherits important techniques from state-of-the-art protocols in the semi-honest setting. Our novel technique to strengthen the protocol for malicious adversaries is inspired by the dual execution technique of Mohassel & Franklin (PKC 2006). Our protocol is optimized for the random-oracle model, but can also be realized (with a performance penalty) in the standard model.

We demonstrate our protocol's practicality with a prototype implementation. To securely compute the intersection of two sets of size 2^{20} requires only 13 seconds with our protocol, which is $\sim 12\times$ faster than the previous best malicious-secure protocol (Rindal & Rosulek, Eurocrypt 2017), and only $3\times$ slower than the best semi-honest protocol (Kolesnikov et al., CCS 2016).

Use Distributed OPRF: as a PSI starting point



to...Shuffled Distributed OPRF



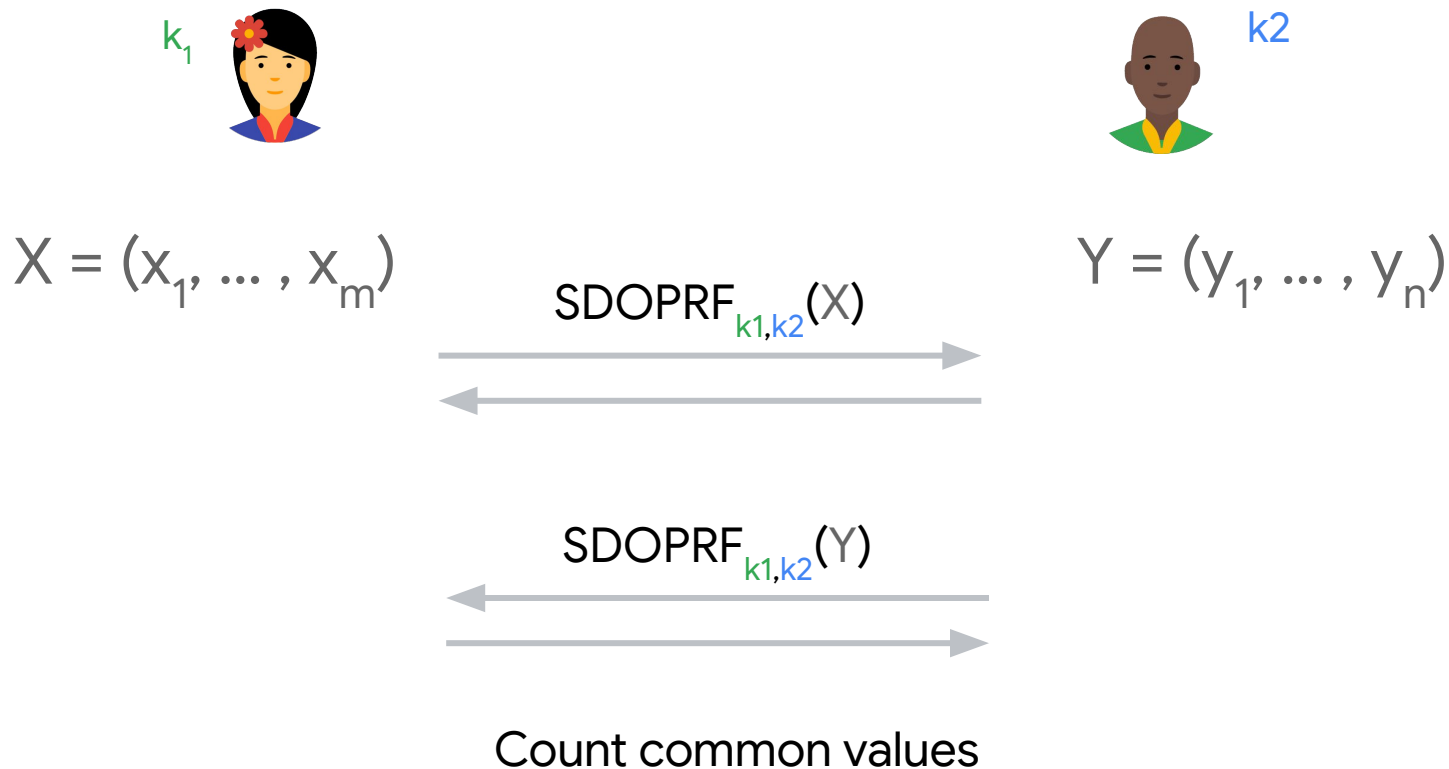
$X = (x_1, \dots, x_m)$



$\text{Shuffle}(\{ F_{k_1, k_2}(x_i) \}_{i \in [m]})$

$\{ F_{k_1, k_2}(x_i) \}_{i \in [m]}$

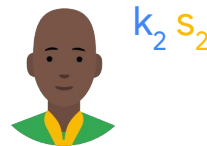
Malicious PSI-Cardinality (two directions)



Malicious PSI-Sum with Cardinality [add HE for sum]



$$X = (x_1, \dots, x_m)$$



$$Y = (y_1, \dots, y_n)$$

$$W = (w_1, \dots, w_n)$$

$$\text{SDOPRF}_{k_1, k_2}(X)$$



$$\text{SDOPRF}_{k_1, k_2}(Y), \text{HEnc}_{s_1, s_2}(W)$$



$$\text{HEnc}_{s_1, s_2}(\text{IntSum})$$

Interactively (and provably) decrypt.

(Avoids a major headache)



Both parties can homomorphically add the encryptions associated with the values in common

What we used? (a lot of technicalities):

- Extended Dodis-Yampolskiy PRF $F_{k_1, k_2}(x) = g^{1/(k_1 + k_2 + x)}$
 - Can be computed interactively (with ZK proofs) by leveraging Camenisch-Shoup (CS) cryptosystem
 - Many Efficient ZK proof of CS, ElGamal, and Strong RSA as a bridge.... (all to do it efficiently)
 - Replacing Sigma-proofs with customized ones
 - Efficient Batching techniques: Damgard-Jurik; Batch OPRF; ElGamal with same first random component!

METHODOLOGICALLY

- We had an **efficient** Honest-but-Curious (a footprint)
- Keep the footprint but change the crypto: **optimize performance**
 - so that you can squeeze max performance out of it
 - (avoid standard ideas: ZK etc., customize for performance!)

To Summarize

- In Theoretical results: well stated problem (well presented/ motivated/ previously unsolved) and a new solution yielding: clever algebra, amazing proof, fundamental techniques, solution to an open question.... is great!!!
- Actual deployment requires: business needs, navigating engineering alternatives, business development, evangelizing, convincing,.. etc.
- Honest design to get “Private Computing”

“practice” in practice!

- One needs to:
 - Insist on best privacy practices whenever possible...
get the best for business needs without violation of
individual data/ individual tracking whenever
possible.... →
 - The Secure Data Exchange is DECISIVELY on the side
of PRIVACY! →
 - Secure Computing between self-secured parties→
Maximizes Privacy and at the same time enables
only Needed Aggregated Utility
- THEN: Science will be needed anyway.....

Beyond the Secure Data Exchange

- Design for scale implies other uses, like “Password check” can be built on it: user checks her password is not in a bad password list without revealing the password (and without learning the list)...
- Other uses...

Crypto in Engineering- general conclusion

- There is **no fixed recipe** for it, just general principles; Business adaptation is challenging; Adaptable efficient methods a win!
- Needed: right interpretation of the theory!
- **Attack models, risk management, incentives** apply, **liabilities** (i.e., legal issues) apply as well.
- Secure components (proofs/ theory) matter!
- **The aesthetics is different than in theory**: solving very real critical valuable issue!

FINAL THOUGHT: theory vs. practice →



The Elegance of Theory

“The elegance of a mathematical theorem is directly proportional to the number of independent ideas one can see in the theorem and inversely proportional to the effort it takes to see them.”

— **George Pólya, Mathematical Discovery on Understanding, Learning, and Teaching Problem Solving, Volume I**

The Elegance of Practice

“If you are out to describe the truth, leave elegance to the tailor.” — **Ludwig Boltzmann**



In working on Actual Solutions

I say, from the perspective of Industrial Research:

“The Technical Problem Solving Process of Highly Messy Real Business Situations/ Needs which seems hopeless is, in fact, an interesting navigation transforming “Hopelessness→ Solution,”
Hence, by definition: It Is Elegant”

THANKS!
