An overview of secure multiparty computation and its application to digital asset custody

Authors
Matt O'Grady, MSc (Royal Holloway, 2021)
Carlos Cid, ISG, Royal Holloway

Abstract

Secure multiparty computation (MPC) is a branch of cryptography that can be used by two or more parties to jointly compute the output of an arbitrary function, without sacrificing the privacy of their respective inputs. MPC has existed since the early 1980s, however interest in the field has experienced a marked increase in recent years, particularly due to its potential to facilitate the secure custody of digital assets such as bitcoin. With the continued adoption of both MPC and digital assets, it is now necessary for security practitioners to be familiar with at least the fundamental concepts underpinning both technologies. As such, this article provides a brief overview of MPC, and further highlights the benefits of MPC-based bitcoin custody over traditional approaches.[a]

[a]This article is published online by Computer Weekly as part of the 2022 Royal Holloway information security thesis series `https://www.computerweekly.com/ehandbook/Royal-Holloway-Secure-multiparty-computation-and-its-application-to-digital-asset-custody`. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full thesis is published on the ISG's website at `https://www.royalholloway.ac.uk/research-and-teaching/departments-and-schools/information-security/research/explore-our-research/isg-technical-reports/`.

Introduction

Consider the following scenario: two curious millionaires wish to determine who, between them, is the wealthiest. Both millionaires are extremely secretive and distrustful, therefore do not want to share their net worth with each other directly, nor allow a mutually trusted third party to perform the comparison on their behalf. This seemingly abstract problem is now famous within the subfield of cryptography known as secure multiparty computation (MPC), first defined by Andrew Yao in 1982, which aims to solve the following problem:

Assume there exist m parties, $P_1, \ldots, P_m$, each of whom possess a private value, $x_1, \ldots, x_m$, respectively. Without the need for any participant to reveal their private value, and without outsourcing the computation to a third party, is it possible for the participants to jointly determine the result of an arbitrary function, $f(x_1, \ldots, x_m)$?

The millionaires' problem can be seen as a concrete example of the general problem described above, where two parties wish to execute a comparison function over each private net worth, with a view to determine which value is greater without revealing these values to each other nor outsourcing the computation to a third party. While the millionaires' problem is a relatively simplistic application of MPC, it has been mathematically proven that *any* arbitrary function can be securely computed[1], meaning there is scope for the technology to be applied to a potentially endless number of real-world problems. Many academics, however, dismissed the concept of MPC, believing it to be a mere theoretical curiosity, as progress towards practical applications stalled for decades due to the inefficiency of solutions that were initially proposed. It was not until the turn of the 21st century that MPC matured to become an efficient tool that could be utilised for many practical purposes.

[1]S. Micali, O. Goldreich, and A. Wigderson, "How to play any mental game," in Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. ACM, 1987, pp. 218–229.

Additive secret sharing

Secure multiparty computation can be achieved using a wide range of mathematical techniques, with early research within the field focussing on the combination of two concepts, namely *oblivious transfer* and *garbled circuits*. Many practical applications today, however, utilise a technique known as *secret sharing*. Broadly, this term is used to describe the act of distributing a secret value, S, between m individual parties, $P_1, \ldots, P_m$, such that each party cannot recover *any* information about S, without knowledge of the values (or shares) possessed by the other participants.

The most straightforward secret sharing scheme, known as an *additive* secret sharing, requires a third-party dealer to define the secret value, S, as the sum of randomly chosen shares. These shares are then individually assigned to each participant and kept secret. To recover the secret, all participants must disclose their share to one another and calculate their sum.

To illustrate this, consider a dealer that wishes to share the secret, $S = 21$, between three participants, Alice, Bob and Charlie. To share S, the dealer must generate three cryptographically secure random values that sum to S – for example, $S_A = 7$, $S_B = 8$ and $S_C = 6$ – and give S_A to Alice, S_B to Bob and S_C to Charlie respectively. The secret, S, is then subsequently deleted by the dealer. With access only to their respective shares, no single participant can determine the value of S, unless they retrieve the shares possessed by all other participants. From the perspective of Alice, this involves asking Bob and Charlie for their shares ($S_B = 8$ and $S_C = 6$, respectively) and calculating the following:

$$S = S_A + S_B + S_C = 7 + 8 + 6 = 21$$

For several reasons, secret sharing schemes implemented in reality perform mathematical operations over mathematical objects known as *finite fields*, however for our purposes the process can be illustrated sufficiently using standard integers. This includes additive secret sharing, as defined above, and *Shamir secret sharing*, which is defined later in this article.

To facilitate secure multiparty computation of a function, f, additive secret sharing can be used by each participant to generate an alternate mathematical representation of their input, sufficient to mask its true value while preserving the correctness of any operations that are applied by the function. To demonstrate this, we again consider three parties, Alice, Bob and Charlie, who now wish to jointly compute the function, $f(x, y, z) = x + y + z$, where x, y and z are Alice, Bob and Charlie's private input respectively. Assuming that Alice, Bob and Charlie choose inputs of $x = 12$, $y = 15$ and $z = 22$ respectively, they are able to jointly compute the output of $f(x, y, z)$ as follows:

1. All three of Alice, Bob and Charlie must randomly generate three additive shares of their respective inputs. In this example, let's assume each party generates the following additive shares, following the process described earlier:

   Alice: $a_1 = 4, a_2 = 3, a_3 = 5.$
   Bob: $b_1 = 5, b_2 = 8, b_3 = 2.$
   Charlie: $c_1 = 19, c_2 = 2, c_3 = 1.$

2. Each participant keeps one share for themselves, and securely sends a share to all other participants. As a result, each participant is in possession of the following shares:

   Alice: $a_1 = 4, b_1 = 5, c_1 = 19.$
   Bob: $a_2 = 3, b_2 = 8, c_2 = 2.$
   Charlie: $a_3 = 5, b_3 = 2, c_3 = 1.$

3. Each participant now computes $f(a, b, c)$, where a is a share of the input generated by Alice, b is a share of the input generated by Bob, and c is a share of the input generated by Charlie, as follows:

Alice: $f(a_1, b_1, c_1) = 4 + 5 + 19 = 28.$
Bob: $f(a_2, b_2, c_2) = 3 + 8 + 2 = 13.$
Charlie: $f(a_3, b_3, c_3) = 5 + 2 + 1 = 8.$

It follows that the result of each computation above is an additive share of the output of $f(x, y, z)$.

4. Finally, the output of f can be recovered as per the additive secret sharing procedure previously described. That is, each participant must disclose their share to all other participants, and calculate their sum as follows:

$$f(28, 13, 8) = 28 + 13 + 8 = 49.$$

As required by the definition of secure multiparty computation, this is identical to the result generated if we have simply calculated $f(12, 15, 22) = 12 + 15 + 22 = 49$, however without the need for any participant to reveal their individual input, nor outsource the computation to a third party. The function outlined in this example was chosen to be relatively simple for demonstration purposes, however secret sharing schemes can also support secure function evaluation involving multiplication operations (e.g. a function such as $f(x, y, z) = xyz$).

Shamir secret sharing

An alternative secret sharing scheme, known as *Shamir secret sharing*, is also commonly utilised to facilitate secure multiparty computation. The construction of Shamir secret sharing is fundamentally the same as additive secret sharing, as the process is similarly comprised of three steps: share generation, distribution, and reconstruction. However, Shamir secret sharing is more flexible, as a secret S can be shared between n parties, such that only a party in possession of t (called a threshold) or more shares is able to recover S. This is commonly known as a t-out-of-n secret sharing scheme, and contrasts additive secret sharing, as the latter requires the cooperation of all participants during the reconstruction phase.

Polynomials

A polynomial $f(x)$ is a sum of several terms that contain different powers of x, and the degree of $f(x)$ is the greatest power of x in the expression. For example, $f(x) = 8 + 11x$ has degree 1, and $f(x) = 1 - x + x^2$ has degree 2.

To share a secret between n participants with a threshold of t, Shamir secret sharing requires a polynomial, $f(x)$, to be defined by the dealer with the following properties:

- It must have a degree of $t-1$ (i.e., the greatest power of x in $f(x)$ must be $t-1$)
- All coefficients must be cryptographically secure random integers
- The secret to be shared must be defined as the point where the polynomial's graph crosses the y-axis (i.e., the value $f(0)$).

Once a polynomial is constructed as above, the dealer can sample n random points on the graph of $f(x)$, which act as shares of the secret and can subsequently be distributed to all participants. The dealer can securely delete the polynomial once all participants have received a share, and the secret can then only be recovered with at least t shares using polynomial interpolation. In essence, interpolation is a tool used to recover a unique polynomial given only a set of points that lie upon its graph and relies on the mathematical fact that a polynomial of degree m requires at least $m + 1$ points to describe it uniquely.

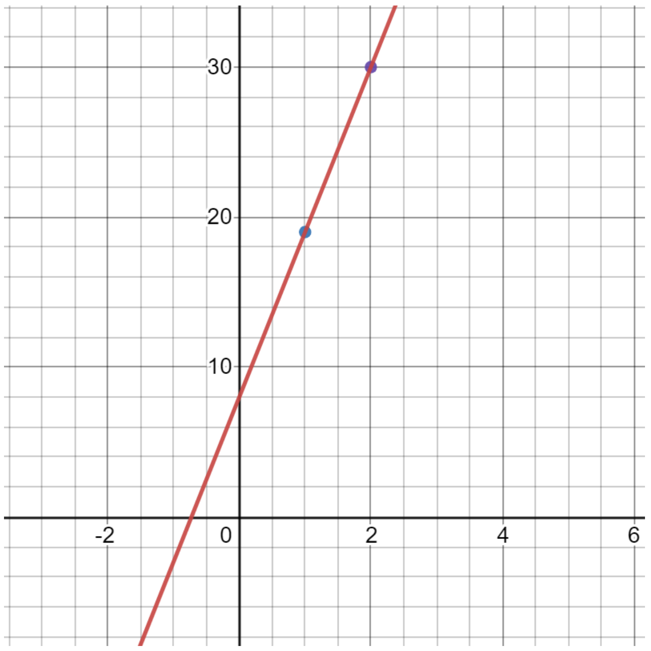

Figure 1: Graph of $f(x) = 8 + 11x$, passing through the points $(1, 19)$ and $(2, 30)$. Clearly given these two points, we can recover the secret $S = 8$ by noting the y-intercept of the graph.

As an example, consider a dealer wishing to share a secret $S = 8$ between three parties, such that at least two must cooperate to recover the secret. This could be shared using the function $f(x) = 8+11x$, and three points on its graph – say, $(1, 19), (2, 30)$ and $(3, 41)$. Here, the coefficient of x (in this case, 11) is a cryptographically secure random integer chosen by the dealer. As $f(x)$ has a degree of one, only two points are required to define it uniquely. Therefore, given two points, the polynomial can be recovered using interpolation, and the secret can subsequently be recovered by finding the y-intercept of the graph (i.e., by evaluating $f(x)$ at $x = 0$). Figure 1 illustrates this.

Conversely, given fewer than two point from this graph, it is mathematically impossible to recover $f(x)$ and subsequently S, as a minimum of two points are required to uniquely define the corresponding polynomial. (See Figure 2.) As a result, no single participant is able to recover S independently.

Utilising Shamir secret sharing instead of additive secret sharing to facilitate secure computation can be beneficial in certain contexts. If just one party is unable (or simply refuses) to disclose their share of the function output when additive secret sharing is used, the final output will no longer be recoverable. Swapping an additive scheme with, say, a t-out-of-n Shamir scheme would allow the final step of the secure function computation to be undertaken with the cooperation of just t participants. As a result, Shamir secret sharing is favoured over additive secret sharing in many modern applications of MPC.

Threshold signatures and digital asset custody

A recent use-case of secure multiparty computation has arisen within digital signatures, particularly when used to facilitate secure custody of digital assets such as bitcoin. A digital signature scheme is a class of cryptographic primitive used to provide both origin authentication and non-repudiation, and is comprised of three algorithms: key generation, message signing and message verification. Digital signatures schemes are used on the bitcoin network to authorise the transfer of bitcoin under the ownership of one user to another. In particular, any user on the network can generate a public/private key pair, both of which have a distinct purpose, and are stored within a bitcoin wallet. A wallet is a software implementation used to interact with the bitcoin network to sign and therefore authorise transactions. The private key authorises a transaction when it is used to produce a signature over the

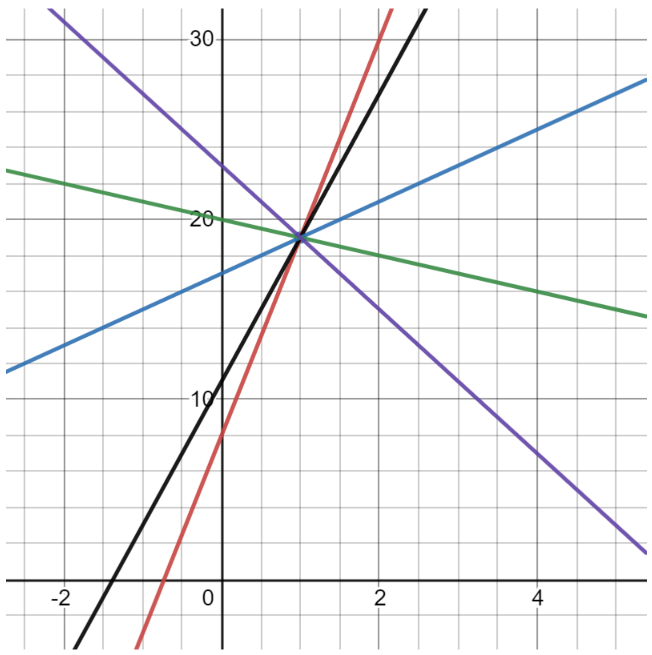

Figure 2: Given a single point, $(1, 19)$, no unique graph can be constructed, therefore S cannot be recovered with only this information.

details of a transaction, such as recipient address and amount. The associated public key is used by nodes on the bitcoin network to verify that the signature was produced by the owner of the associated private key, hence whether they are authorised to spend the bitcoin specified in the transaction.

An inherent weakness related to key management arises when bitcoin transactions are authorised in this way. A survey conducted in 2016 by Krombholz et al. found that out of 990 participants, 22.5% admitted to having lost the private key associated with a bitcoin wallet under their ownership at least once in the past[2]. The reasons cited include user error (such as hard drive formatting or a misplaced private key), security breaches resulting in theft, hardware failures, and software failures. Indeed, the most prominent and heart-breaking example of this is the story of James Howell, who mistakenly threw away a hard drive containing a private key associated with 7500 bitcoin[3], which at the time of writing is worth over £246m. Primarily, this issue occurs as a result of a single point of failure introduced during key storage – if a user loses their private key, which is simply a single 256-bit integer, the associated bitcoin are irretrievable.

t-out-of-n threshold signature

At least t signatories are required to produce a valid signature.

Fortunately, threshold signature schemes have the capacity to mitigate this. A threshold signature scheme is an extension of a traditional signature scheme that allows multiple users to participate in the signature generation process. This is unlike traditional signature schemes that can only produce a signature on behalf of a single entity. Threshold signature schemes utilise secure multiparty computation techniques (e.g., secret sharing schemes) to allow the signing parties to define a threshold of participants at the key generation phase, that must be present during the signing phase in order to produce a valid signature. Any number of signatories less than this defined threshold will not be able to produce a valid signature. Applying this to the key storage weakness highlighted earlier, if a 2-out-of-3 threshold signature scheme was used to authorise bitcoin transactions, as long as at most one of three keys is lost or stolen, the associated bitcoin would remain

[2]K. Krombholz, A. Judmayer, M. Gusenbauer, and E. Weippl, "The other side of the coin: User experiences with bitcoin security and privacy," in International conference on financial cryptography and data security. Springer, 2016, pp. 555–580.

[3]https://www.theguardian.com/uk-news/2021/jan/14/man-newport-council-50m-helps-find-bitcoins-landfill-james-howells [Accessed: Feb-2022]

accessible. Furthermore, threshold signatures allow for joint custody of bitcoin, whereby a threshold of participants (i.e., two in the example above) must agree to authorise a bitcoin transaction. This clearly has many beneficial applications in the real world when funds are shared between parties.

Alternative solutions

A more standard approach taken to reduce the likelihood of a single point of failure in many systems involves backups. A bitcoin seed phrase is a list of 12 to 24 words (typically written on paper or steel) used to backup and recover a bitcoin private key in the event of its loss or accidental destruction. Threshold signatures, in theory, reduce the necessity of seed phrases, as a level of redundancy is introduced when the threshold is defined. For example, if a 3-out-of-5 threshold signature scheme was used to authorise bitcoin transactions, as long as no more than two shares are lost, the bitcoin remain accessible.

However, a seed phrase backup does not make private key theft any less likely. In a way, having one or more backups of a private key in the form of a seed phrase increases its attack surface, as these backups require adequate physical protection if they are stored on paper or steel following common practice. On the other hand, threshold signatures are particularly useful at protecting private keys from theft. This is because a threshold private key is produced in a distributed manner, such that each party contributes to the generation process, but no single party is ever in possession of the private key in its entirety. The private key is also never reconstructed or stored in a central location, even when a message is being signed. It is in this sense that threshold signature schemes can be seen as an application of secure multiparty computation, where the key generation and signing algorithms are functions that multiple parties wish to jointly compute, without a trusted third-party nor the requirement for any parties' input (i.e., their contribution to the private key) to be revealed to any other participant. As a result, a private key associated with a threshold wallet is much more difficult to steal than a single centralised key stored in a standard wallet, as it requires a malicious entity to breach a threshold of shares simultaneously in order to mount a successful attack.

Readers familiar with the bitcoin codebase could argue that this functionality has already been implemented natively, and so may question the need for threshold signatures. This implementation is often referred to as *multisig*, and enables the same threshold transaction approval properties (albeit without utilising MPC) as described above. However, utilising threshold signatures offers the following additional benefits:

- Reduced transaction fees – A multisig bitcoin wallet with a threshold of t is essentially comprised of a n distinct wallets, each associated with their own public and private key. To authorise a transaction from a set of multisig wallets, at least t of n participants must sign it separately with their respective private keys. This means the size of the transaction (in bytes) scales in proportion to size of both n and t. As network usage fees depend on the number of bytes in a transaction, utilising native multisig can become costly if either t or n (or both) are large. A bitcoin wallet implementing a threshold signature scheme circumvents this issue, because the output of the signing protocol is a single signature, and as a result the size of a transaction remains constant despite the size of n and t.
- Increased privacy – As a threshold signature scheme produces a single signature as output, an external party observing the blockchain would be unable to distinguish between a transaction signed by a single party and one signed by multiple parties. This is in contrast to native bitcoin multisig, which requires each transaction to contain at least t signatures along with the public key of all n owners of the multisig wallet, potentially revealing the identity of those involved in the transaction.

The future

Despite the continued growth and adoption of digital assets, the extent to which threshold signatures have been implemented to secure them is still limited. Three of the most well-known companies currently working in this space are Sepior, UnboundSecurity and ZenGo. Of the companies listed, however, only ZenGo currently offer a consumer-grade threshold signature wallet, hugely restricting the exposure the general public have to this ground-breaking technology.

Fortunately, this is expected to change dramatically over the coming years, following Coinbase's acquisition of UnboundSecurity, who now expect to roll out MPC capabilities across a number of their consumer-focussed digital asset products and services[4]. Although we are still a number of years away from witnessing the mass adoption and implementation of threshold signature schemes within digital asset wallets, given the benefits of MPC to this application, it is unlikely that interest and innovation in this field will be slowing down any time soon.

Biographies
Matt O'Grady graduated from Royal Holloway, University of London in 2021 with an MSc in Information Security, where he was awarded the prize of Best ISG Student Overall (2020/21) for achieving the highest grade average within the cohort. Prior to this, he graduated from the University of Exeter, with a first-class degree in Mathematics. Currently, Matt works as an Information Security Consultant at Accenture, providing security advice and guidance to clients in both the public and financial sector.

Carlos Cid is a Professor in the Information Security Group at Royal Holloway University of London, and a Visiting Research Professor at Simula UiB, Bergen, Norway. His main research interests are cryptography and cyber economics. He has a PhD in Pure Mathematics from the University of Brasilia, Brazil (1999), and before joining Royal Holloway in 2003, held academic and industry roles in Brazil, Germany and Ireland. Carlos was the founding director of Royal Holloway's Centre for Doctoral Training in Cyber Security, a post he held between 2013 and 2017.

Series editor: S.- L. Ng

[4]https://blog.coinbase.com/coinbase-to-acquire-leading-cryptographic-security-company-unbound-security-9c7e67f37e24 [Accessed: Feb-2022]