# Technical Report on Decentralized Multifactor Authentication

Nillion

November 28, 2023

**Abstract**

With the contemporary surge in the storage and transmission of high-value digital data, like medical records and financial transactions, securing that data is more crucial than ever. To mitigate the risks associated with centralized systems, decentralized approaches distribute this data across different servers in various trust domains. However, the benefits of this approach cannot be realized if the mechanisms that govern access to data are still centralized. This document dives into decentralized authentication's challenges, emphasizing the necessity for robust multi-factor authentication (MFA). It reviews legal aspects by drawing from top standards, sets the context by exploring a range of cryptographic techniques at a high level, and focuses in detail on two novel multi-party computation (MPC) protocols for MFA. Finally, it addresses engineering challenges associated with decentralizing widely used factors (such as email and phone verification) and embracing wallet ownership validation.

## 1 Introduction

More and more types of sensitive and high-value data, such as medical records, financial transactions and genome material, are being stored digitally. In order to avoid a central point of failure, this information can be distributed over $n$ servers hosted in different trust domains, so that a malicious attacker would have to compromise at least several of them to get access to the data. Accessing and using this data then requires a decentralized authentication protocol.

Traditional authentication systems face inherent challenges that arise from their centralized nature. Authentication against a central server introduces a single point of failure and requires trust in this centralized entity.

Web3 is an open web built on decentralized technology such as blockchains. In Web3, centralized servers are no longer the linchpin of authentication. Instead, users typically authenticate to a blockchain by proving ownership of the private key of a wallet. This decentralized approach eliminates the need to trust a central server, offering a resilient and trust-minimized authentication process with no single point of failure. However, the shift to a decentralized architecture introduces a gap in multifactor authentication (MFA) [54, 40]. Recognized as a best practice, MFA involves the use of several authentication factors, categorized into four types: (1) something you know, such as a password, (2) something you have, like a unique identifier from a mobile device, (3) something you are, such as a facial biometric, and (4) something you do [54]. The proof of possession of a single factor in Web3, the private key, lacks the additional security provided by the combination of different factors in MFA. For example, should the private key be lost or stolen, the absence of a mechanism to regain control of the account poses a significant challenge.

Designing an MFA decentralized solution is an ongoing challenge since some of the traditional MFA factors are hard to decentralize. For example, email verification (something you have) typically

involves a central server sending an email with a link to the user, but this is a centralized process that is not easy to decentralize. Moreover, given the sensitive nature of MFA information, we need a decentralized authentication solution that preserves data confidentiality, which suggests the use of privacy enhancing technologies (PET).

Zero-knowledge proofs (ZKP) are one of the most prevalent PETs used in current authentication solutions as they seamlessly adapt to the client-server architecture with its prover/verifier structure, enabling users to prove to a *verifier* server possession of authentication factors without compromising sensitive information [56]. But they need to be adapted to be used in the context of a decentralized architecture. Indeed, the verifier server, which constitutes a single-point-of-failure and requires trust, needs to be replaced by a network of servers. The distributed essence of multi-party computation (MPC) naturally accounts for this without requiting protocol modifications [26]. In addition, most MPC protocols can be proven to be secure in the UC framework [14], which allows combining MPC protocols that verify different factors into one secure protocol that verifies their combination without leaking intermediary results. For these reasons, in this report we focus on MPC-based solutions in order to address the general question of how to support MFA on a decentralized infrastructure.

# 2 Related Work

In this section, we explore the application of relevant privacy-enhancing technologies for authentication purposes, with a specific focus on decentralized authentication.

## 2.1 ZKP

A zero-knowledge proof is a cryptographic technique allowing one entity (the *prover*) to demonstrate knowledge of a specific value to another entity (the *verifier*) without divulging the actual value. An illustration of this is proving that you know a password without disclosing it. A more advanced example is proving that your fingerprint at login time is the same (i.e. similar enough) to the one that was registered during enrollment without disclosing any of the two. These two examples illustrate the use of ZKP for exact [47] and approximate matching [68, 35], respectively (see Section 4).

ZKPs, like other privacy-enhancing technologies, offer protection against eavesdropping-based attacks such as man-in-the-middle (MiTM), IP spoofing, denial of service (DoS), and replay attacks when data is transmitted over an untrusted network. In a client-server (i.e. centralized) architecture, ZKPs are seamlessly implemented with the client serving as the prover and the server as the verifier. By distributing the verifier across a network of nodes, authentication against decentralized networks, such as blockchains, becomes achievable using ZKP techniques [50, 1, 59, 62]. For example, Decentralized Identifiers (DIDs) [63] are a foundational component of decentralized identity systems, providing a way to create and manage globally unique identifiers that are not tied to a central authority, for instance on a blockchain. Verifiable Credentials leverage DIDs to enable secure and privacy-preserving sharing of digital credentials [64]. The integration of Zero Knowledge Proofs (ZKPs) in the context of Verifiable Credentials aims at enhancing privacy by allowing individuals to prove the validity of their credentials without revealing the underlying sensitive information, contributing to a more secure and user-centric digital identity ecosystem. However, at least two noteworthy challenges arise in practice when applying ZKP to decentralized authentication.

Firstly, not all ZKPs are secure under composition [49], which is crucial for combining multiple protocols in multi-factor authentication (MFA) in a safe way. Composability within the Universal Composability (UC) framework [14, 15] is highly emphasized in pragmatic cryptography engineering for several compelling reasons. It serves as the recommended best practice due to its role in ensuring

that cryptographic building blocks seamlessly integrate without interference, both within a single protocol and when concurrently used in different instances. The adoption of universally composable (UC-secure) Non-Interactive Zero-Knowledge (NIZK) protocols becomes crucial in designing larger cryptographic systems [38, 48], offering a "worry-free" approach akin to using "ideal boxes". Despite its theoretical foundation, UC is of vital importance in practical cryptography, providing protection against numerous perils and subtle attacks that may arise when composability is overlooked. The adherence to composability principles, especially within the UC framework, facilitates the robustness and security of cryptographic systems in real-world applications.

Secondly, When decentralized networks, particularly blockchains, are involved, additional challenges emerge due to regulatory compliance issues arising from the immutability of blockchain records. Indeed, the unalterable nature of blockchain entries complicates the correction of errors or updates in credential status. Furthermore, the public visibility of blockchain data may clash with regulatory mandates, such as the rights to erasure and restriction of processing, whereas immutability could hinder the exercise of rights such as the right to rectification and data portability among others [30, 36].

### 2.2 Biometric hashing

Biometric hashing is a technology akin to conventional cryptographic hashing, serving as a one-way function to transform biometric data into an irreversible cryptographic hash instead of a biometric template. Contrary to cryptographic hash functions, where close inputs are not mapped to close outputs, most biometric hash functions establish a correlation between hash codes in the hash space and the similarities among biometric samples in the original space using techniques such as matrix mapping or learning-based methods. This correlation is called a *relation preserving* property [75]. This way, as long as the presented biometric data during login is *similar* to the one submitted during user enrollment, the obtained hashes are also similar. Due to its irreversibility, low computational cost, and high storage efficiency, biometric hashing is an interesting technique in privacy-preserving biometric recognition systems [60, 65, 43]. However, the relation preserving property requires striking a balance between the system's ability to hash similar inputs to similar outputs and the underlying security of the hash function. In recent years, many reconstruction attacks based on exploiting the relation preserving property have appeared [67, 29, 27, 44], allowing the attackers to reconstruct the underlying biometric information from the binary hash codes.

### 2.3 HE

Homomorphic encryption (HE) is an encryption scheme allowing operations between the message and ciphertext spaces. Depending on the scheme, it may support homomorphic addition ($a + b = \mathsf{Dec}(\mathsf{Enc}(a) + \mathsf{Enc}(b))$) and/or homomorphic multiplication ($a \times b = \mathsf{Dec}(\mathsf{Enc}(a) \times \mathsf{Enc}(b))$), where $a$ and $b$ are messages, $\mathsf{Enc}$ and $\mathsf{Dec}$ are encryption and decryption algorithms, and the operations on ciphertexts may differ from those on plaintexts. Partially homomorphic encryption (PHE) supports unlimited operations of one type [55], somewhat homomorphic encryption (SHE) supports a limited number of operations for both types [12], and fully homomorphic encryption (FHE) supports an unlimited number of operations for both types [31].

HE techniques have been mostly employed to cover biometric authentication factors: Face recognition [61], iris recognition [53], fingerprint recognition [73], voice recognition [57], signature recognition [32], multi-modal factors [70]. Besides the great amount of research effort, HE has been mostly employed in the client-server model, where, in summary, the client provides encrypted information about the authentication factor which is then compared with the encrypted information

held in the server. All thirty works analysed in this recent review [72] only address this two-party case.

A key security challenge inherent in the client-server model, specifically in centralized Homomorphic Encryption (HE) solutions, revolves around the handling of cryptographic keys. Typically, HE encryption employs a public key for encryption and assigns the responsibility of safeguarding the corresponding secret key for decryption to the client. This places a significant burden on the client, who is often not a security expert and tends to be the more vulnerable party. Consequently, the risk of theft or loss of the private key is increased. Additionally, as discussed in [72, Section 3.5], HE may suffer potential attacks ranging from side-channel attacks, lattice attacks and others.

In terms of performance, Homomorphic Encryption (HE) schemes demand substantial computing resources due to their inherent high computational complexity. Notably, certain biometric authentication methods utilizing HE exceed a 10-second duration [72]. It is important to highlight that authentication time has to adhere internet-wide timeouts to prevent session closure. For instance, Google's products, such as Gmail's SMTP server with a timeout of 10 seconds [66], and Google's content delivery networks with a default timeout of 5 seconds extended to 15 [33], exemplify cases where connection timeouts play a crucial role in system responsiveness.

### 2.4 Trusted Execution Environments

A Trusted Execution Environment (TEE) is an environment where the code executed and the data accessed are isolated and protected by a secure enclave. TEEs are increasingly common and are also used in the context of biometric authentication; this includes Apple's Touch ID and Face ID [3], BTAA which is a blockchain and TEE-assisted authentication [52] and TEEBAG which requests the remote device to send the stored authentication templates to the TEE. [7]. However, commercial implementations of TEEs have several important vulnerabilities across existing systems. First, critical implementation bugs such as buffer overflows inside Trusted Applications (TAs) [17]. Second, architectural deficiencies such as ill-implemented memory protection mechanisms [28]. Third, overlooked hardware properties such as side channels make TEEs vulnerable to attacks such as cache-timing attacks, as processors are very complex [34].

### 2.5 MPC

In an ideal decentralized environment involving $n$ servers, a secure yet less efficient approach requires a client to authenticate themselves individually with each server using $M$ multiple authentication factors. This simple method poses a trade-off between usability and security. Secure Multiparty Computation (MPC) [18, 10, 21] emerges as a promising technology to overcome this inherent dilemma. MPC serves as a privacy-centric solution with the capacity to introduce decentralization, addressing the challenges posed by conventional methods. This innovative primitive enables a group of parties to collaboratively evaluate a predefined function while ensuring the privacy of their respective inputs.

Traditionally, the focus of MPC systems has predominantly revolved around addressing privacy concerns related to authentication, with a notable emphasis on the two-party setting, where server-client authentication is paramount [13, 74]. Two-party secure computation protocols, especially those exploring biometric factors such as fingerprints [39] utilizing FingerCode representation [42] and facial recognition [58] employing EigenFaces [69], have garnered significant attention. As reported in [13], secure fingerprint authentication takes approximately 1 second, while secure facial recognition authentication averages around 6.5 seconds. Other works delve into expanding the type of authentication factors, facilitating secure multi-modal biometric authentication. For instance,

SEMBA [9] seamlessly integrates information from face and iris modalities. Unlike earlier approaches leveraging garbled circuit techniques [10], SEMBA implements the MPC framework through the SPDZ protocol [21], yielding secure authentication mechanisms with an online runtime ranging from 30 ms to 120 ms. It is crucial to note that these timings, however, do not account for communication and round complexity, as the client and server are co-located on the same machine.

To harness the decentralization potential of MPC technology and remove the authentication burden from the user without compromising security, one can adopt insights from [4, Section 6.2] regarding authentication methods. The study [4] outlines the applicability of MPC for both password-based and biometric authentication. In these scenarios, MPC facilitates a secure comparison between the user-provided password/feature vector and its encrypted counterpart on the server. Extending this concept to a distributed environment involves utilizing a secret-shared password/feature vector across $n$ servers, necessitating only one interaction between the client and servers per authentication factor $M$.

In a parallel research trajectory, MPCAuth [66] exploits the decentralization feature of MPC to achieve a similar outcome: Enabling a client and a consortium of $n$ servers to emulate the client's authentication as if directed to a single logical server. This innovative approach employs TLS as an MPC protocol and applies it to various authentication factors, including email, SMS, universal second factor, biometrics, and security questions. MPCAuth not only introduces the concept but also provides an implementation of its core protocol (TLS-in-SMPC protocol) alongside the associated authentication mechanisms. They assume a round-trip time of 20 ms and, for email authentication, assuming an established TLS-in-SMPC connection, they report an offline runtime of 3.69 s and an online runtime of 430 ms in a 5 server setting. The end-to-end system for both TLS-in-SMPC connection and authentication has an offline runtime of 18.52 s and an online runtime of 1.71 s in the same 5 server setting.

## 3 Multi-Factor Authentication

Several regulations mandate or recommend the use of multifactor authentication (MFA) across various industries to enhance security. Some prominent regulations include

- The Payment Services Directive 2 (PSD2): Applies to financial institutions in the European Union and requires strong customer authentication (SCA) [6] for electronic payments [24].
- General Data Protection Regulation (GDPR): Applies to organizations handling personal data of EU citizens and encourages the use of robust security measures, including MFA, to protect personal information [30].
- Health Insurance Portability and Accountability Act (HIPAA): Applies to healthcare organizations in the United States, emphasizing the need for secure access controls, including MFA, to protect patients' health information [37].
- Banking: Regulatory initiatives in various countries, such as the anti-money laundering (AML) directives [25] or the UK's Open Banking, require financial institutions to implement SCA for secure access to financial services.

Alongside these regulations, several standards set guidelines and best practices around the use of MFA. Two prominent ones are NIST SP 800-63 - Digital Identity Guidelines [54] and ISO/IEC 29115 – Entity Authentication Assurance Framework (EAAF) [40]. Both standards describe several levels of assurance (LoA) indicating the degree of confidence in the identity or credential. For example,

NIST SP 800-63 contemplates three different LoAs that specifically address the strength of the authentication process used to verify the identity of the individual: AAL1 (Little or No Confidence) is a single-factor authentication, typically using a password, AAL2 (Some Confidence) involves two factors, and AAL3 (High Confidence) requires MFA, with higher assurance in the authentication process. NIST SP 800-63 contemplates four different types of authentication factors:

- Something You Know: This includes knowledge-based factors such as passwords, personal identification numbers (PINs), or answers to security questions.
- Something You Have: This includes possession-based factors such as a physical token, a smart card, or a mobile device.
- Something You Are: This includes biometric factors such as fingerprints, voice recognition, retina scans, or facial recognition.
- Something You Do: This includes behavioral factors such as typing patterns, or other behavioral biometrics.

The selection and number of authentication factors for a given application depend on factors such as the application's risk tolerance, the sensitivity of the data involved, and the user experience requirements. Finding equilibrium between security and usability is crucial, and aspects such as the potential impact of a security breach and the user population's familiarity with specific authentication methods should be taken into account in the decision-making process

NIST SP 800-63 and ISO/IEC 29115 describe three essential phases: Enrollment, credential management and authentication. Enrollment involves user onboarding, identity proofing (see ISO/IEC Technical Specification 29003 - Identity Proofing [41]), and, upon successful completion, the creation of a user account and entry in a register. Credential Management includes overseeing the life-cycle of a credential for a registered user, encompassing creation, issuance, activation, usage, revocation, and destruction/archiving. Authentication pertains to the operational use of a credential by the user.

In this report, we focus on three specific steps within the three phases of enrollment, authentication, and credential management:

1. Account registration within the enrollment phase: User account registration results in the creation of a user account and credentials. These credentials typically include at least a User Id and password but can encompass additional factors or attributes.
2. Login within the authentication phase: Returning users undergo returning user authentication (RUA), presenting and validating different credentials according to the risk model.
3. Change factor within the credential management phase: The user is able to change one credential (i.e. a factor) by relying on the successful verification of other existing credentials.

All the steps described above need to take into account the following security considerations. First, they need to be supported by a decentralized network of nodes where no node acts as a leader or authority. Second, they need to preserve confidentiality of the underlying credentials, so that they are not revealed to any number of colluding nodes below a threshold $t$. Third, they need to employ one-time randomness to mitigate risks such as man-in-the-middle (MITM) and replay attacks, and make use of information-theoretic cryptography in order to prevent brute force attacks. Additional desired requirements for the underlying technology are the preservation of data integrity, availability and resilience of processing.

# 4 Exact and Approximate Matching for MFA via MPC

Most operations on MFA factors involve exact or approximate matching. Exact matching can be used for instance to support password authentication, or to check if a user's device identifier is correct. Approximate matching can be used to support different types of biometric authentication, based for example on fingerprints, voice recognition, retina scans, or facial recognition. In this section we discuss the MPC approach to these two types of matching.

Exact matching determines whether $x = y$ for secrets $x, y$. In some instances such as password authentication, one of the two secrets has already been stored in the network at registration during enrollment time. At the MPC level this last detail makes little difference, so in Subsection 4.2 we discuss the general problem of exact matching instead.

Approximate matching determines whether two vectors are sufficiently close, according to some distance function $d(\cdot, \cdot)$. Once more, in some cases one of the vectors has already been stored in the network during enrollment time, but we abstract ourselves from this implementation detail and present the general problem of approximate matching instead. This is discussed further in Subsection 4.3.

Passive security in MPC involves defending against adversaries who corrupt participants without altering the protocol, often referred to as honest-but-curious or semi-honest adversaries, whereas active security extends protection to adversaries capable of arbitrarily deviating from the protocol, potentially modifying their code to exploit vulnerabilities, also known as malicious or Byzantine adversaries. Active adversaries in MPC pose a greater threat by having the capability to modify the output of a computation, introducing an additional layer of potential manipulation. For this reason, we focus on efficiency for active security in the dishonest majority setting. In any case, we're assuming that the underlying LSSS-based MPC protocol is capable of securely generating sharings of random numbers, outputting them to the client and computing multiplications. In this case, state-of-the-art MPC protocols for online multiplication, used in both exact and approximate matching, require at least one round [21, 45, 46]. In order to eliminate these online rounds we make use of the protocol described in the next section.

## 4.1 The Underlying MPC Protocol

Fast user authentication is crucial for a seamless user experience, enhancing user satisfaction and efficiency. Minimizing wait times ensures users can access the desired services promptly, contributing to overall system usability and productivity. For this reason, we make use of the MPC protocol introduced in [71] in order to minimize user wait times. This protocol demonstrates an optimal online phase in terms of communication rounds, referring to the operations requiring user presence. In essence, it minimizes the user wait time by requiring communication only for sending inputs and receiving computation results, without necessitating network nodes to exchange messages during the computation, thereby optimizing the overall protocol execution time.

Protocol [71] introduces the concept of a *masked factor* $\langle x \rangle_\lambda$ hiding a non-zero secret $x \in \mathbb{Z}_p^*$ with independent uniformly random mask exponent $\lambda \in \mathbb{Z}_{p-1}$: It is given as $\langle x \rangle_\lambda := x \cdot g^{-\lambda} \in \mathbb{Z}_p^*$, where $g$ is a public generator in $\mathbb{Z}_p^*$. This protocol can evaluate an arbitrary (multivariate) polynomial with optimal online round complexity. In its pre-processing phase, it requires for each monomial the computation of a correlation pair, which requires exponentiating a public generator $g$ by a secret-shared exponent $\gamma$. Each one of these can be computed with passive security with an exponentiation protocol [2] requiring $n - 1$ fan-in 2 multiplications.

Both the exact matching and the initial component of the approximate matching problem can be expressed in terms of a polynomial. We notice that in both problems, the fan-in of the products that

arise in every monomial of the polynomial is low (i.e. the degree of the monomials is low). Hence instead of pre-computing sharings $[g^\gamma]$ using the exponentiation protocol, we do pre-processing by multiplying invertible random values, reducing the number of pre-processed multiplications. Specifically, the protocol for a monomial $[x_1^{j_1} \cdots x_k^{j_k}]$ is as follows: Precompute a product of random numbers $[r_1^{j_1} \cdots r_k^{j_k}]$, optionally check if these numbers are all invertible, and let the clients submit $\langle x_i \rangle := r_i^{-1} x_i$. The nodes then locally compute the product

$$[x_1^{j_1} \cdots x_k^{j_k}] = \langle x_1 \rangle^{j_1} \cdots \langle x_k \rangle^{j_k} [r_1 \cdots r_k].$$

The general claim for an arbitrary polynomial then works by extending linearly.

### 4.2 Exact matching

In order to determine whether two secrets $x$ and $y$ are the same, it suffices to determine whether $x - y$ is nonzero. Naively revealing the value $x - y$ leaks information on $x$ and $y$ however, so instead the value $r(x - y)$ for $r$ a nonzero random number will be computed. Ordinarily, the approach in a linear secret sharing scheme (LSSS)-based MPC system would then be as follows:

**Exact match in one round,** $\pi_{\texttt{ExactMatchOneRound}}$.

$$[r(x-y)] \leftarrow \pi_{\texttt{ExactMatchOneRound}}(x, y)$$

Preprocessing:

1. The nodes compute a sharing $[r]$ of a random value $r$.
2. They determine whether $r$ might be zero; if so, restart. Usually the approach, dating back to [8] is: Multiply by another random value $r'$ and reveal the result.

Online:

3. The clients send $[x]$ and $[y]$ to the nodes (one or both of them possibly long in advance, so retrieved from storage instead).
4. The nodes locally compute the sharing $[x - y] = [x] - [y]$ using linearity of the LSSS.
5. They then compute the multiplication $[r(x - y)] = [r] \cdot [x - y]$.

Ordinarily the cost would hence be as follows: 2 random values and 1 public multiplication [16] in preprocessing, and 1 multiplication in the online phase. By making use of the MPC protocol in Subsection 4.1, we can get rid of the online multiplication with a bit of pre-processing as follows, under the very reasonable assumption that both $x$ and $y$ are nonzero:

**Exact match in zero rounds,** $\pi_{\texttt{ExactMatchZeroRounds}}$.

$$[r(x-y)] \leftarrow \pi_{\texttt{ExactMatchZeroRounds}}(x, y)$$

Preprocessing:

1. The nodes compute sharings $[r], [r_x], [r_y], [r'], [r'']$ of five random values $r, r_x, r_y, r', r''$.
2. They multiply $[r_x r] = [r_x] \cdot [r]$, then use a public multiplication to compute $r_x \cdot r \cdot r'$. If this value is zero, restart. Similarly, they multiply $[r_y r] = [r_y] \cdot [r]$, then use a public multiplication to compute $r_y \cdot r \cdot r''$. If zero, restart.
3. The nodes send $[r_x]$ to the input client of $x$ and $[r_y]$ to the input client of $y$, who reconstruct $r_x$ and $r_y$ respectively.

Online:

3. The clients broadcast $\langle x \rangle := r_x^{-1} x$ and $\langle y \rangle := r_y^{-1} y$ to the nodes (one or both of them possibly in advance, and retrieved from storage instead).
4. The nodes locally compute the sharing $[r(x-y)] = \langle x \rangle [r_x r] - \langle y \rangle [r_y r]$ using linearity of the LSSS.

Preprocessing cost is thus 5 random values, 2 multiplications and 2 public multiplications, and no multiplications in the online phase are required.

**Remark 4.1.** *The first public multiplications can be skipped in the last protocol, the trade-off being that a small probability arises (in preprocessing) that the client complains that the value $r_x$ is zero and it needs to be remade.*

### 4.3 Approximate matching

For approximate matching, the aim is to compute a sharing of the inequality $[d(x, y) < T]$, where $x$,$y$ are secret shared values coming from clients or storage, $d(\cdot, \cdot)$ is some distance function and $T$ is a threshold which is usually allowed to be public but may also be kept secret if preferred.

In the context of biometrics, the most common choice for $d(\cdot, \cdot)$ is the Euclidean distance; equivalently one may consider its square, which is easier in the setting of MPC. Other choices might include the cosine, Mahanoblis and Manhattan similarity distances.

LSSS protocols in the dishonest majority setting ordinarily require an extra round of communication to compute the squared Euclidean distance $d(\cdot, \cdot)$, which we can remove as before. Naively doing this for the polynomial $\sum_{i=1}^{\ell} z_i^2$ where each $z_i$ will be equal to $x_i - y_i$ is problematic, as it would leak whether $x_i = y_i$. Instead, we will ask the clients to secret-share $\sum_{i=1}^{\ell} x_i^2$ and $\sum_{i=1}^{\ell} y_i^2$, and compute $-2 \sum_{i=1}^{\ell} x_i y_i$ as before. To ensure that nothing is leaked in case $x_i$ or $y_i$ is zero, simply shift them both by 1; this does not affect $d(\mathbf{x}, \mathbf{y})$.[1]

[1] In order to prevent overflows, the unshifted inputs $x_i$ should be less than $\sqrt{(p-1)/\ell}$, which can be ensured with zero-knowledge proofs.

**Euclidean distance squared in zero rounds,** $\pi_{\texttt{EDSZeroRounds}}$.

$$[d(\mathbf{x},\mathbf{y})] \leftarrow \pi_{\texttt{EDSZeroRounds}}(\mathbf{x},\mathbf{y})$$

Preprocessing:

1. The nodes compute sharings $[r_{x_i}], [r_{y_i}]$ of $2\ell$ random values $r_{x_i}, r_{y_i}$.
2. They perform $\ell$ multiplications $[r_{x_i} r_{y_i}] = [r_{x_i}] \cdot [r_{y_i}]$.
3. The nodes send the $[r_{x_i}]$ to the input client of $\mathbf{x}$ and the $[r_{y_i}]$ to the input client of $\mathbf{y}$, who reconstruct the values $r_{x_i}$ and $r_{y_i}$ respectively.

Online:

3. The clients broadcast $\langle x_i \rangle := r_{x_i}^{-1} x_i$ and $\langle y_i \rangle := r_{y_i}^{-1} y_i$ to the nodes, in addition to $[\sum_{i=1}^{\ell} x_i^2]$ and $[\sum_{i=1}^{\ell} y_i^2]$ (as before, potentially in advance).
4. The nodes locally compute the sharing

$$[d(x,y)] = [\sum_{i=1}^{\ell} x_i^2] + [\sum_{i=1}^{\ell} y_i^2] - 2 \sum_{i=1}^{\ell} \langle x_i \rangle \langle y_i \rangle [r_{x_i} r_{y_i}]$$

using linearity of the LSSS.

The pre-processing cost is thus $2\ell$ random values, $\ell$ multiplications, and no multiplications in the online phase are required. As before, if one wants to eliminate the risk of clients complaining that some of the $r_{x_i}$ or $r_{y_i}$ are zero, the protocol is easily modified to use an additional $\ell$ random values and $\ell$ public multiplications to ensure that this is not the case.

Regarding the online computation of the subsequent comparison $[d(x,y) < T]$ (not shown in the protocol above), the Catrina-De Hoogh protocol [16] results in a 3 or 4 online round protocol, depending on the underlying ring.

An attacker might try to trick the comparison protocol by overflowing into a small value; although this is unlikely to work, one could demand zero-knowledge proofs to force the the input vectors lie in a specific range, so no overflows can occur.

# 5 Decentralizing MFA Factors

In this section we adapt the previous MFA concepts to the decentralized case. We focus on three steps (account registration, returning user authentication and change factor) within the three phases of enrollment, authentication and credential management contemplated in NIST SP 800-63 and ISO/IEC 29115.

**Account Registration.** During account registration, often referred to as *sign up*, the user registers credentials that fall into one or more NIST SP 800-63 authentication factor types. The main difference compared to a centralized solution, is that the authentication factors are not stored in a single server, but in a network of nodes. Also, in an MPC-based solution the authentication factors are not sent in plaintext to the network nodes, but rather using some form of threshold-based privacy-preserving mechanism (e.g. secret shares in LSSS MPC [21, 20, 22], encryption in FHE

MPC [19], or encoding in garbled circuits MPC [11, 51]). The threshold $t$ determines the maximum number of network nodes that may collude without being able to reconstruct an authentication factor. The main feature of MPC is that it allows the network nodes to work with the authentication factors without ever having to reconstruct them, for example during returning user authentication.

**Returning User Authentication.** This function, often referred to as *login* or *sign in*, requires the user to verify a number of factors of different types in order to prove that they are the same user that created the account. In a decentralized solution, this verification is done against each one of the network nodes, so that they can independently conclude that the user is the same. In the MPC case, since factors are stored by the network using some privacy-preserving mechanism, verifying a factor requires running an MPC protocol. In many cases, the purpose of this protocol is to determine whether the credential presented during registration and login are exactly the same. This is the case for instance for factors such as passwords and device identifiers. In some cases, more complex computations are required. For example, in biometric authentication an MPC protocol is typically required to compute the distance between the registration and login factors and to compare its result against a threshold. Since the MPC protocols we propose are secure in the UC framework [14], we can combine different exact and approximate matching protocols into a single secure MPC protocol. This way, we do not leak the partial result obtained from each factor match, but only the final *pass/fail* result.

**Changing Factors.** Users might want to change an authentication factor for a number of reasons, including a new, lost, or stolen device, a forgotten password, to increase security, or to comply with regulations. Best practices recommend changing one factor at a time, and doing so by authenticating through a number $n$ of other factors, where $n$ depends on the trade-off between risk tolerance and user experience. In order to minimize risk, the selected $n$ factors should cover as many different NIST SP 800-63 authentication types as possible.

### 5.1 Biometric Authentication

Biometric authentication validates a user's identity by leveraging distinctive biological attributes like fingerprints, voice, retina, and facial features. The essence of biometric authentication lies in securely storing this unique biological information, allowing for robust identity verification whenever a user seeks access to their account. If a password is hacked, a new one can be generated, whereas this is not the case for biometric information, which is therefore considered as very sensitive data. For this purpose, and to minimize the risk of this information being hacked or used by a central authority without our permission, it makes sense to store it in a secure decentralied network.

The standard biometric authentication process unfolds in two distinct stages: Initial biometric data capture on the client side and subsequent verification within the network. Initially, the client captures diverse biometric data, such as fingerprints, iris scan, or facial features. This data undergoes encoding such as through a neural network, which transforms it into a logical representation called a *feature vector* or a *biometric template*. This is in turn translated into a fixed-point rational representation compatible with finite field operations. The encoded information is then securely shared across the network using some form of information hiding mechanism, such as a linear secret sharing scheme. Subsequently, the network calculates the Euclidean distance between the stored biometric feature vector from the registration phase and the biometric feature vector uploaded during the login attempt utilizing an MPC protocol so that the biometric information is never reconstructed. The MPC protocol also compares this distance to a predetermined threshold, declaring the login successful if the distance falls below the specified threshold. Notably, the network only discerns the

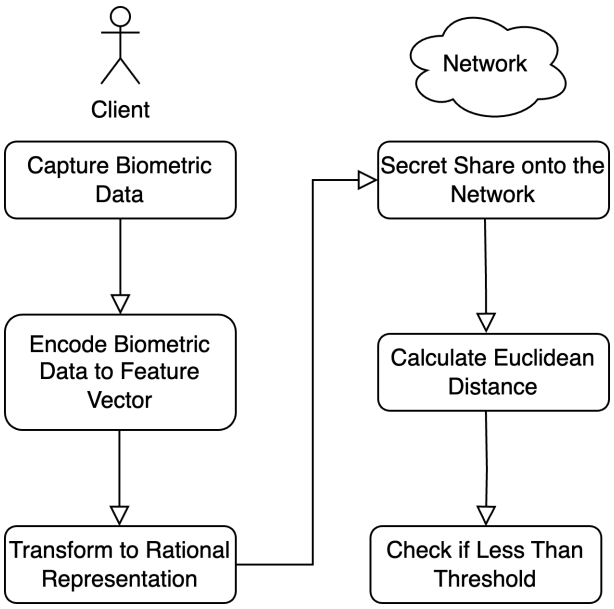


Figure 1: Secure decentralized biometric authentication flow.

success or failure of the login attempt without gaining access to specific biometric details.

### 5.2 Email and Phone

Traditional authentication methods often rely on email and phone validation as additional layers of verification to enhance security. A server sends an email or a text message with a code or a link to the user so that the latter can prove that they control the corresponding email address or phone number. While these methods are straightforward to implement in a client-server (i.e. centralized) architecture, they are not easy to port to a decentralized architecture. The key challenge here is how to minimize the trust placed on the node sending the email or text message, while maintaining a good user experience (UX) that is consistent with what users are familiar with.

For example, a naive way to decentralize the action of sending a verification email would be to have each node in the network send a separate email to the user with a link, and to require the user to click on that link in each email. This UX is neither satisfactory nor consistent with what users are familiar with.

The solution detailed in Figure 2, complies with best practices on UX while exhibiting *some* of the properties one would get from a decentralized network. This solution requires a network of servers, each one with the ability to send a verification email or text message with a link to the user. Following a decentralized random process (or the user's choice), a node $A$ is selected. This node $A$ sends a verification message with a link to the user, receives an HTTP GET request when the user clicks on the link and broadcasts this information to the rest of the network.

The decentralized process selecting node $A \in \{0, 1, ..., n-1\}$ at random can be a simple MPC protocol with the following steps: 1) The $n$ nodes generate a secret sharing $[a]$ of an unknown random element $a \in \mathbb{F}_p$ using the standard ideal functionality $F_{\texttt{RAN}}$ [23], 2) $a$ is publicly revealed, 3) each node locally computes $A \equiv a \pmod{n}$.

This solution removes the server sending the email as a single point of failure since now a timeout can trigger the random selection of another node $A'$. Moreover, an actor performing a denial of service attack does not know the identity of the next node $A$, effectively limiting the efficacy of their

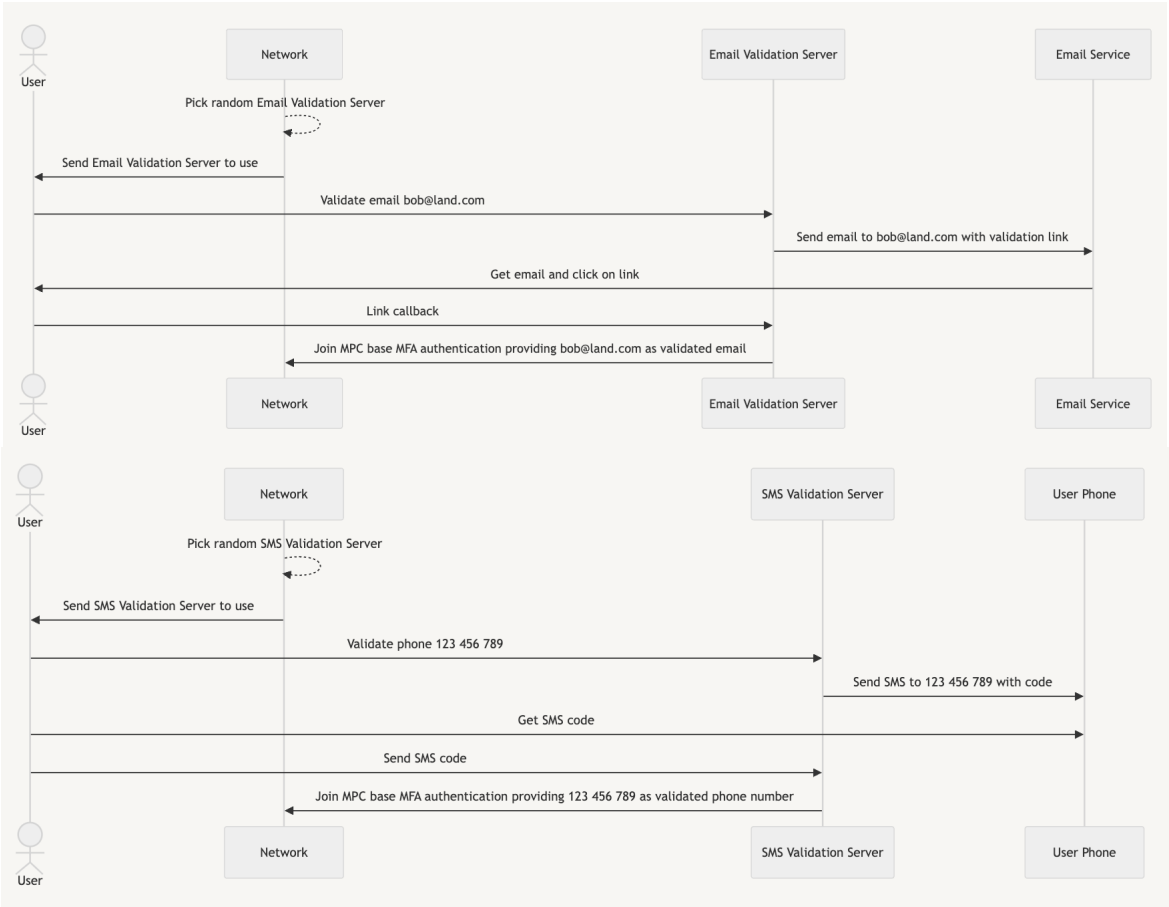


Figure 2: Email (top) and phone (bottom) verification flows.

attack. Therefore, this solution is more fault tolerant than the traditional one. However, it still requires placing a certain amount of trust on the nodes sending the emails. For instance, it does not describe a cryptographic way for the other nodes to verify that they have indeed sent the correct email to the customer [2]. However, users can complain if the process does not work. In order to align node incentives, staking can be used and this type of conflict resolution can be combined with *slashing*.

### 5.3 Wallets

Wallets are natural authentication factor to be chosen in a decentralized solution. What follows is a flow proposal to authenticate against the network using Ethereum wallets. This description is valid for both regular Externally Owned Accounts (EOAs, regular Ethereum wallets) and the new wallet accounts based on ERC-4337, in what follows referred to as a *managed wallet*.

There will be different situations depending on where the Ethereum wallet comes from:

- Pre-existing wallet: The user already owns a wallet and wants to use it as an additional factor to authenticate with the network.

[2]Notice that requiring the sender node to send an email to the user putting the other nodes in copy allows them to click on the link instead of the user, increasing the attack surface.

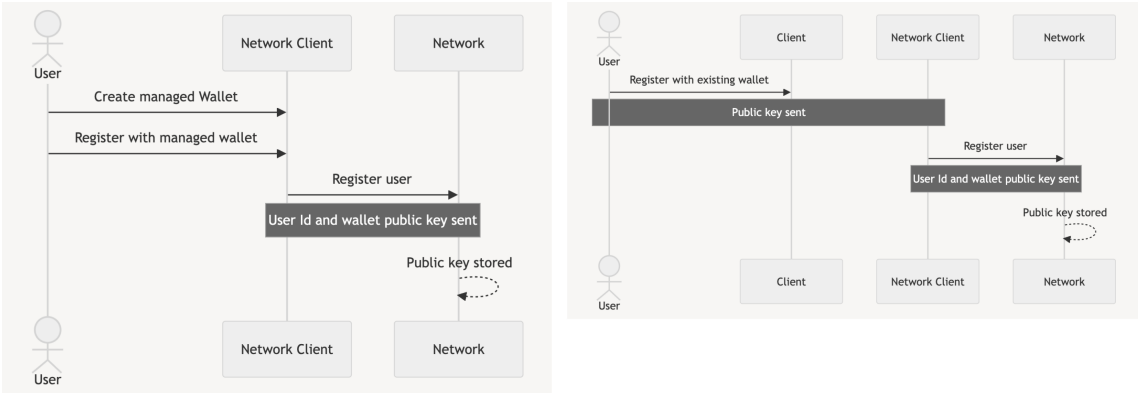


Figure 3: Wallet registration flows for a pre-existing (left) and a managed wallet (right).

- Wallet created and managed by the network client on behalf of the user: While registering with the network the user creates a new wallet exclusively for interacting with the network (using different wallets for different purposes is a security best practice)

During registration of an existing or new managed wallet (see Figure 3), the public key is sent to the network nodes, which store it linked to the User Id. In the managed wallet case the wallets could be automatically registered after creation.

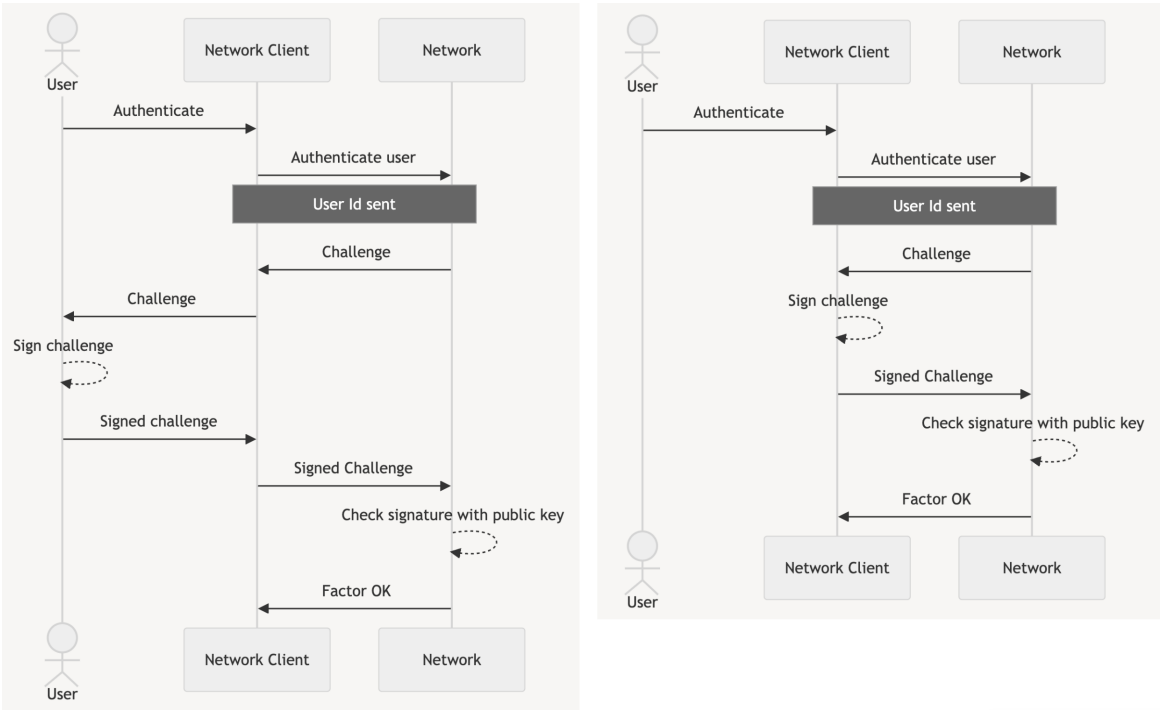


Figure 4: Wallet authentication flows for a pre-existing (left) and a managed wallet (right).

When the user is authenticating to the network with their User Id (see Figure 4), the latter generates a random number through a decentralized protocol (see *challenge*) and sends it back to the user. The user signs with its private key a message containing the challenge and sends it to the network nodes, which check both that the challenge is correct and that the signature corresponds to

the public key of associated to the User Id. A different challenge is used per authentication in order to prevent replay attacks.

### 5.4 Other Factors

Next to traditional factors we can consider ubiquitous factors that might lead to multi-factor authentication which might be very hard to spoof. For example, consider a future landscape of authentication [5], where additional factors seamlessly integrate into the user experience without imposing additional friction. Picture a system that contemporaneously incorporates geolocation, Wi-Fi network identifiers, Wi-Fi signal strength thresholds, and device IDs from nearby devices, such as smartwatches, smartphones, laptops, smart cards, and other devices. This passive multi-factor authentication serves as an augmentative layer to enhance security.

Now, consider the dynamic nature of these indicators throughout a typical workday. The signature at home differs from that in transit or at the office, providing a nuanced representation of identity. Leveraging the variability in these indicators over time allows for identity representation or access controls based on this contextual richness.

Taking a progressive step, introduce a social factor into the authentication paradigm. For instance, daily proximity to a consistent coworker, whose smartphone, smartwatch, and laptop also contribute to passive multi-factors, further enriches the authentication process through near-field communication between devices.

This low-threshold, non-interactive multi-factor authentication comprises distinct components: A master mesh encompassing personal devices like smartwatches, smartphones, laptops, and smart cards, an environmental mesh embracing Wi-Fi routers, network printers, domotics, network-attached storage, and Bluetooth car radios, and a behavioral mesh incorporating time (morning, noon, evening) and location (home, transit, office, exception).

Addressing concerns about deviations from routine, a degradation of service principle, linked to access controls, provides a solution. For example, during work-related travel with a standard set of personal devices and a known coworker, dynamic access controls might restrict access to critical documents while permitting access to other resources. This can be complemented by incorporating active factors, offering alternative authentication methods like facial recognition.

Overall, the augmented authentication factors contemplated is this scenario would require decentralized exact or approximated matching protocols as the ones detailed in Section 5.

## 6 Conclusions

Multi-factor authentication (MFA) is considered a best practice in authentication, recommended by standards and prescribed by many regulations. However, its implementation within a decentralized architecture presents various challenges. This report thoroughly examines these challenges, covering the regulatory legal framework and incorporating best practices derived from prominent standards. A comprehensive exploration of cryptographic aspects includes an up-to-date analysis of various techniques. The report scrutinizes two multi-party computation (MPC) protocols for exact and approximate matching, foundational for critical MFA operations. Engineering hurdles related to migrating factors to a decentralized framework, such as email and phone verification, are discussed. Additionally, the assimilation of other factors, such as validating wallet ownership, or combining device, environmental and behavioral information is explored.

## References


[1] Onais Ahmad et al. "BAuth-ZKP—A Blockchain-Based Multi-Factor Authentication Mechanism for Securing Smart Cities". In: *Sensors* 23 (Mar. 2023), p. 2757.

[2] Abdelrahaman Aly, Aysajan Abidin, and Svetla Nikova. *Practically Efficient Secure Distributed Exponentiation without Bit-Decomposition*. Cryptology ePrint Archive, Paper 2019/334. https://eprint.iacr.org/2019/334. 2019.

[3] Apple. "iOS Security". In: *Technical Report May, Apple Inc* (2016).

[4] David W Archer et al. "From Keys to Databases—Real-World Applications of Secure Multi-Party Computation". In: *The Computer Journal* (Sept. 2018). ISSN: 1460-2067.

[5] Charles Aunger et al. "Enhanced authentication techniques using virtual persona". Patent US20220070166A1. Pending, filed on August 27, 2021, by Health2047 Inc. Priority to application US17/446,210. Published on March 3, 2022. Assignee: Health2047 Inc. 2022.

[6] Financial Conduct Authority. *PS21/19: Changes to the SCA-RTS and to guidance in the Approach Document and the Perimeter Guidance Manual*. Retrieved 1 February 2021. The Financial Conduct Authority.

[7] Ranjbar A. Balisane and Andrew Martin. "Trusted Execution Environment-Based Authentication Gauge (TEEBAG)". In: *NSPW '16: Proceedings of the 2016 New Security Paradigms Workshop* (Sept. 2016), pp. 61–67.

[8] J. Bar-Ilan and D. Beaver. "Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction". In: *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*. PODC '89. Edmonton, Alberta, Canada: Association for Computing Machinery, 1989, pp. 201–209. ISBN: 0897913264.

[9] Mauro Barni et al. "SEMBA: secure multi-biometric authentication". In: *IET Biometrics* 8.6 (Aug. 2019), pp. 411–421. ISSN: 2047-4946.

[10] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. "Foundations of garbled circuits". In: *Proceedings of the 2012 ACM conference on Computer and communications security*. CCS'12. ACM, Oct. 2012.

[11] Osman Biçer. *Efficiency Optimizations on Yao's Garbled Circuits and Their Practical Applications*. 2017. arXiv: 1703.03473 [cs.CR].

[12] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 325–341. ISBN: 9783540305767.

[13] Julien Bringer, Herve Chabanne, and Alain Patey. "Privacy-Preserving Biometric Identification Using Secure Multiparty Computation: An Overview and Recent Trends". In: *IEEE Signal Processing Magazine* 30.2 (2013), pp. 42–52.

[14] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *J. Cryptol.* 13.1 (Jan. 2000), pp. 143–202. ISSN: 0933-2790.

[15] Ran Canetti. "Studies in secure multiparty computation and applications". In: 1995.

[16] Octavian Catrina and Sebastiaan De Hoogh. "Improved Primitives for Secure Multiparty Integer Computation". In: SCN'10. Amalfi, Italy: Springer-Verlag, 2010, pp. 182–199. ISBN: 364215316X.

[17] David Cerdeira et al. “SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems”. In: *2020 IEEE Symposium on Security and Privacy* (2020).

[18] David Chaum, Claude Crépeau, and Ivan Damgard. “Multiparty Unconditionally Secure Protocols”. In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC ’88. Chicago, Illinois, USA: Association for Computing Machinery, 1988, pp. 11–19. ISBN: 0897912640.

[19] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. “Multiparty Computation from Threshold Homomorphic Encryption”. In: *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. EUROCRYPT ’01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 280–299. ISBN: 3540420703.

[20] Ivan Damgård and Yuval Ishai. “Scalable Secure Multiparty Computation”. In: *Proceedings of the 26th Annual International Conference on Advances in Cryptology*. CRYPTO’06. Santa Barbara, California: Springer-Verlag, 2006, pp. 501–520. ISBN: 3540374329.

[21] Ivan Damgård et al. “Multiparty Computation from Somewhat Homomorphic Encryption”. In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662. ISBN: 978-3-642-32009-5.

[22] Ivan Damgård et al. “Scalable Multiparty Computation with Nearly Optimal Work and Resilience”. In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 241–261. ISBN: 978-3-540-85174-5.

[23] Ivan Damgård et al. “Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation”. In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 285–304. ISBN: 978-3-540-32732-5.

[24] “Directive (EU) 2015/2366 of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market, amending Directives 2002/65/EC, 2009/110/EC and 2013/36/EU and Regulation (EU) No 1093/2010, and repealing Directive 2007/64/EC (Text with EEA relevance)”. In: OJ L (). Retrieved 12 July 2020.

[25] *Directive (EU) 2018/843 of the European Parliament and of the Council of 30 May 2018 amending Directive (EU) 2015/849 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing, and amending Directives 2009/138/EC and 2013/36/EU (Text with EEA relevance)*. In force. URL: http://data.europa.eu/eli/dir/2018/843/oj.

[26] David Evans, Vladimir Kolesnikov, and Mike Rosulek. 2018.

[27] Yi Feng, Meng Hui Lim, and P C Yuen. “Masquerade attack on transform-based binary-template protection based on perceptron learning”. In: *Pattern Recognition* 47 (Sept. 2014), pp. 3019–3033.

[28] Fabian Fleisher, Marcel Busch, and Phillip Kuhrt. “Memory corruption attacks within Android TEEs: a case study based on OP-TEE”. In: *ARES ’20: Proceedings of the 15th International Conference on Availability, Reliability and Security* 53 (Aug. 2020), pp. 1–9.

[29] Javier Galbally et al. “On the Vulnerability of Face Verification Systems to Hill-Climbing Attacks”. In: *Pattern Recogn.* 43.3 (Mar. 2010), pp. 1027–1038. ISSN: 0031-3203.

[30] GDPR Info. *Regulation (EU) 2016/679 (General Data Protection Regulation)*. URL: https://gdpr-info.eu/.

[31] Craig Gentry. "A Fully Homomorphic Encryption Scheme". AAI3382729. PhD thesis. Stanford, CA, USA, 2009. ISBN: 9781109444506.

[32] Marta Gomez-Barrero et al. "Implementation of Fixed-Length Template Protection Based on Homomorphic Encryption with Application to Signature Biometrics". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2016, pp. 259–266.

[33] Google Cloud. *Cloud CDN Quotas*. Accessed November 28, 2023. n.d. URL: https://cloud.google.com/media-cdn/quotas.

[34] Johannes Gotzfried et al. "Cache Attacks on Intel SGX". In: *EuroSec'17: Proceedings of the 10th European Workshop on Systems Security* 2 (Apr. 2017), pp. 1–6.

[35] Chunjie Guo, Lin You, and Gengran Hu. "A Novel Biometric Identification Scheme Based on Zero-Knowledge Succinct Noninteractive Argument of Knowledge". In: *Security and Communication Networks* 2022 (Sept. 2022), pp. 1–13.

[36] Harry Halpin. *A Critique of Immunity Passports and W3C Decentralized Identifiers*. 2020. arXiv: 2012.00136 [cs.CR].

[37] *Health Insurance Portability and Accountability Act (HIPAA)*. URL: https://www.govinfo.gov/app/details/CRPT-104hrpt736/CRPT-104hrpt736.

[38] S. Hohenberger et al. "Anonize: A large-scale anonymous survey system". In: *2014 IEEE Symposium on Security and Privacy*. IEEE. 2014, pp. 423–438.

[39] Yan Huang et al. "Efficient Privacy-Preserving Biometric Identification." In: Jan. 2011.

[40] *ISO/IEC 29115:2013 - Information technology – Security techniques – Entity authentication assurance framework*. International Organization for Standardization (ISO). URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:29115:ed-1:v1:en.

[41] *ISO/IEC TS 29003:2018 - Information technology – Security techniques – Identity proofing*. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC). URL: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:ts:29003:ed-1:v1:en.

[42] A.K. Jain et al. "FingerCode: a filterbank for fingerprint representation and matching". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. CVPR-99. IEEE Comput. Soc.

[43] R. Megiba Jasmine and J. Jasper. "A Privacy Preserving Based Multi-Biometric System for Secure Identification in Cloud Environment". In: *Neural Process. Lett.* 54.1 (Feb. 2022), pp. 303–325. ISSN: 1370-4621.

[44] Emre Kaplan et al. "Known Sample Attacks on Relation Preserving Data Transformations". In: *IEEE Transactions on Dependable and Secure Computing* PP (Oct. 2017), pp. 1–1.

[45] Marcel Keller. "MP-SPDZ: A Versatile Framework for Multi-Party Computation". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1575–1590. ISBN: 9781450370899.

[46] Marcel Keller, Valerio Pastro, and Dragos Rotaru. "Overdrive: Making SPDZ Great Again". In: Jan. 2018, pp. 158–189. ISBN: 978-3-319-78371-0.

[47] Nesrine Khernane, Maria Potop-Butucaru, and Claude Chaudet. "BANZKP: A secure authentication scheme using zero knowledge proof for WBANs". In: *2016 13th International Conference on New Technologies for Distributed Systems (NOTERE)*. 2016, pp. 1–6.

[48] A. Kosba et al. "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 839–858.

[49] Ahmed Kosba et al. *C∅C∅: A Framework for Building Composable Zero-Knowledge Proofs*. Cryptology ePrint Archive, Paper 2015/1093. https://eprint.iacr.org/2015/1093. 2015.

[50] Wanxin Li et al. "Aggregated Zero-Knowledge Proof and Blockchain-Empowered Authentication for Autonomous Truck Platooning". In: *IEEE Transactions on Intelligent Transportation Systems* 24.9 (Sept. 2023), pp. 9309–9323. ISSN: 1558-0016.

[51] Yehuda Lindell et al. *Efficient Constant Round Multi-Party Computation Combining BMR and SPDZ*. Cryptology ePrint Archive, Paper 2015/523. https://eprint.iacr.org/2015/523. 2015.

[52] Wenze Mao, Peng Jiang, and Liehuang Zhu. "BTAA: Blockchain and TEE-Assisted Authentication for IoT Systems". In: *IEEE Internet of Things Journal* 10 (14 2023).

[53] Mahesh Kumar Morampudi et al. "Secure and verifiable iris authentication system using fully homomorphic encryption". In: *Computers amp; Electrical Engineering* 89 (Jan. 2021), p. 106924. ISSN: 0045-7906.

[54] *NIST Special Publication 800-63-3: Digital Identity Guidelines*. National Institute of Standards and Technology (NIST). URL: https://pages.nist.gov/800-63-3/sp800-63-3.html.

[55] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *International Conference on the Theory and Application of Cryptographic Techniques*. 1999.

[56] Adwait Pathak et al. "Secure Authentication using Zero Knowledge Proof". In: *2021 Asian Conference on Innovation in Technology (ASIANCON)*. 2021, pp. 1–8.

[57] Yogachandran Rahulamathavan. *Privacy-preserving Similarity Calculation of Speaker Features Using Fully Homomorphic Encryption*. 2022.

[58] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. "Efficient Privacy-Preserving Face Recognition". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 229–244. ISBN: 9783642144233.

[59] Xavier Salleras, Sergi Rovira, and Vanesa Daza. "FORT: Right-Proving and Attribute-Blinding Self-Sovereign Authentication". In: *Mathematics* 10.4 (Feb. 2022), p. 617. ISSN: 2227-7390.

[60] Tobias Scheidat, Claus Vielhauer, and Jana Dittmann. "Biometric hash generation and user authentication based on handwriting using secure sketches". In: *2009 Proceedings of 6th International Symposium on Image and Signal Processing and Analysis*. 2009, pp. 89–94.

[61] Hatef Otroshi Shahreza et al. "Hybrid Protection of Biometric Templates by Combining Homomorphic Encryption and Cancelable Biometrics". In: *2022 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, Oct. 2022.

[62] Lihua Song et al. "An Access Control Model for the Internet of Things Based on Zero-Knowledge Token and Blockchain". In: *EURASIP J. Wirel. Commun. Netw.* 2021.1 (Apr. 2021). ISSN: 1687-1472.

[63] Manu Sporny, Amy Guy, Markus Sabadello, et al. *Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations*. W3C Recommendation 19 July 2022. Latest published version: https://www.w3.org/TR/did-core/. World Wide Web Consortium (W3C), July 2022.

[64] Manu Sporny, Dave Longley, David Chadwick, et al. *Verifiable Credentials Data Model v1.1*. W3C Recommendation 03 March 2022. Latest published version: https://www.w3.org/TR/vc-data-model/. World Wide Web Consortium (W3C), Mar. 2022.

[65] Yagiz Sutcu, Husrev Taha Sencar, and Nasir Memon. "A Secure Biometric Authentication Scheme Based on Robust Hashing". In: *Proceedings of the 7th Workshop on Multimedia and Security*. MMSec '05. New York, NY, USA: Association for Computing Machinery, 2005, pp. 111–116. ISBN: 1595930329.

[66] Sijun Tan et al. "MPCAuth: Multi-factor Authentication for Distributed-trust Systems". In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2023.

[67] Berkay Topcu et al. "Practical security and privacy attacks against biometric hashing using sparse recovery". In: *EURASIP Journal on Advances in Signal Processing* 2016.100 (2016).

[68] Quang Tran et al. "A Privacy-Preserving Biometric Authentication System With Binary Classification in a Zero Knowledge Proof Protocol". In: *IEEE Open Journal of the Computer Society* PP (Dec. 2021), pp. 1–1.

[69] Matthew Turk and Alex Pentland. "Eigenfaces for Recognition". In: *Journal of Cognitive Neuroscience* 3.1 (Jan. 1991), pp. 71–86. ISSN: 1530-8898.

[70] Dilip Kumar Vallabhadas and Mulagala Sandhya. "Securing multimodal biometric template using local random projection and homomorphic encryption". In: *Journal of Information Security and Applications* 70 (Nov. 2022), p. 103339. ISSN: 2214-2126.

[71] Miguel de Vega et al. *Evaluation of Arithmetic Sum-of-Products Expressions in Linear Secret Sharing Schemes with a Non-Interactive Computation Phase*. Cryptology ePrint Archive, Paper 2023/1740. https://eprint.iacr.org/2023/1740. 2023.

[72] Wencheng Yang et al. "A Review of Homomorphic Encryption for Privacy-Preserving Biometrics". In: *Sensors* 23.7 (Mar. 2023), p. 3566. ISSN: 1424-8220.

[73] Wencheng Yang et al. "Secure Fingerprint Authentication with Homomorphic Encryption". In: *2020 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, Nov. 2020.

[74] Yang Yang et al. "A Comprehensive Survey on Secure Outsourced Computation and Its Applications". In: *IEEE Access* 7 (2019), pp. 159426–159465. ISSN: 2169-3536.

[75] L. Yu et al. "Secure biometric hashing against relation-based attacks via maximizing min-entropy". In: *Comput. Secur.* 118 (2022).