Privacy and accountability in identity systems: the best of both worlds

Christian Paquin, Microsoft Research

September 11, 2013

Abstract

Privacy and accountability are widely believed to be opposing goals in identity systems. On one hand, service providers require users to be identifiable to reduce fraud; on the other, users want to limit tracking while minimizing the amount of information disclosed about them. As a result, debates on identity become a rope pulling effort with privacy proponents on one end and security ones on the other. We will illustrate that this opposition is in fact an illusion.

Modern cryptography makes it possible to achieve both strong security and privacy to any degree desired. In this paper, we present a system allowing honest users to access online resources anonymously, but when a user contravenes to the terms of service or acts fraudulently, an auditor can de-anonymize and then ban the misbehaving user from the system. We also describe a prototype using a mobile phone as a second factor of authentication implementing this system.

This paper showcases a new ID escrow system to verifiably encrypt user pseudonyms for an auditor, and an efficient revocation accumulator scheme compatible with the U-Prove technology.

Introduction

As more services are migrated online, system designers are faced with seemingly contradicting requirements. On one hand, they need to provide adequate security for online transactions increasing in value, and simple usernames/passwords do not to provide sufficient protection anymore. On the other hand, privacy requirements protecting against user profiling and tracking are becoming more common and complex.

Many systems prioritize one aspect over the other, and most often than not, security. However, it is now better understood that privacy is an as important aspect that must be considered when designing new systems.[1] As noteworthy example, the White House published last year the National Strategy for Trusted Identities in Cyberspace (NSTIC)[2], outlining the vision of an identity ecosystem that provides both strong privacy and security for consumer identities to be used online.

Multiple technologies are being proposed in the industry to address the privacy issues; but system designers frequently equate privacy with classical cryptographic *secrecy*. Protecting privacy entails more than encrypting user data and attributes, it also involves reducing to the strict required minimum the information learned about the user's activities and data for all parties involved in an identity transaction.

The key is therefore to balance both sets of requirements and design systems that can provide both the required privacy and security protections, using the right mechanisms.

So-called minimal disclosure technologies[3] provide strong cryptographic protections across the privacy spectrum: from anonymity, pseudonymity, to full-identification. For example, U-Prove[4] and Idemix[5] are two predominant technologies backed by important industry players that have been widely studied in academia and in various research projects.[6] They may work well in an ideal world, but reality is more complicated. First, full-anonymity is rarely needed in practice; most identity scenarios require the presentation of some sort of attribute (for example, "I'm pseudonym XYZ", "I'm over-21", "my clearance level is higher than SECRET"). Second, anonymity services routinely get abused, prompting critics to lobby for shutting them down for security reasons. As an example, TOR[7] network nodes are routinely blocked by various organizations and systems to prevent illegal activities, therefore also precluding honest users from its benefits.[8] Privacy services are beneficial to the internet community, but *operational and authorized* privacy services are even more useful!

---

[1] The notion of *privacy by design* is getting more popular and widely advocated.

[2] http://www.nist.gov/nstic/

[3] Also called *anonymous credentials*.

[4] http://www.microsoft.com/u-prove

[5] http://www.zurich.ibm.com/security/idemix/

[6] One noteworthy and currently active project, ABC4Trust, aims at architecting and implementing a common abstraction over U-Prove and Idemix (and any other minimal disclosure technologies), and running real-life pilots with these technologies. https://abc4trust.eu.

[7] https://www.torproject.org/

[8] As an example, Wikipedia does not allow editing pages when using TOR, thus preventing anonymous contributions. Certain sensitive topics would benefit from qualified yet anonymous authors; our system could be used to prove certified qualification of an anonymous author, allowing the anonymity to be lifted by the Wikipedia administrators in case of abuse, and by revoking the author's accreditation.

To enhance their survival chances in the real world, privacy-protecting technologies should therefore offer accountability mechanisms to allow honest participants to use them, while allowing misbehaving ones to be identified and/or banned.

In this paper, we present a system that extends the U-Prove technology to provide an ID escrow service to de-anonymize users committing fraudulent transactions and a revocation mechanism to prevent these users from further accessing the system. We also describe a prototype implementing this system that uses a mobile phone as a second factor of authentication to increase security.

About U-Prove

U-Prove is an innovative cryptographic technology that allows users to minimally disclose certified information about themselves when interacting with online services. U-Prove provides a superset of the security features of Public Key Infrastructure (PKI), and also provides strong privacy protections by offering superior user control and preventing unwanted user tracking. The U-Prove technology specifies cryptographic primitives that can be integrated in the leading federation protocols (like SAML, WS-Trust,[9] OpenId, OAuth, etc.) to enhance their privacy characteristics.

Microsoft released the U-Prove cryptographic specification under the Open Specification Promise[10] allowing anyone to use and implement it freely, and released an open-source SDK[11] implementing it. With the new version 1.1 (revision 2) release of the specification, it is now possible to extend the capabilities of the technology by defining external modules. The ID Escrow and revocation schemes we present later are examples of such extensions.

Academic research in the area of privacy technologies such as U-Prove has been booming in the last few years.[12] Many of the new schemes require more advanced mathematics and rely on newer security assumptions that have not been vetted in practice. Fortunately, U-Prove and the new extension schemes presented herein rely on conventional security assumptions (the same ones as in DSA and ECDSA),[13] and the mathematics are simple enough to be implemented using math libraries widely available.

Two-factor security

Two-factor authentication is an important feature in many identity systems, and is now supported in leading consumer web properties offered by the likes of Microsoft[14] Facebook[15] and Google.[16] Using a dedicated security device or a mobile phone is a good way to increase the confidence in the user

[9] See for example the U-Prove WS-Trust V1.0 profile.
http://research.microsoft.com/apps/pubs/default.aspx?id=166974
[10] http://www.microsoft.com/openspecifications/en/us/programs/osp/security/default.aspx
[11] https://uprovecsharp.codeplex.com/
[12] See for example a recent MAC-based scheme providing multi-show unlinkability and U-Prove like performance: http://eprint.iacr.org/2013/516.
[13] http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
[14] http://www.microsoft.com/en-us/account/default.aspx
[15] http://www.facebook.com/help/270942386330392/
[16] https://support.google.com/accounts/topic/28786

authentication. In most federation protocols however, the user retrieves an authentication statement from an identity provider using a second factor, but the retrieved "bearer" statement is presented as is to a relying party without being linked to the second factor. This makes the statement susceptible to hijacking or replay by an attacker.

A growing number of security devices (from one-time password generators to USB keys performing some cryptography) are now available to act as a second factor of authentication in identity transactions. However, given the cost involved in deploying these devices to users, wide-scale adoption remains a problem. We see a tendency to migrate the schemes to a device that most users already have in their pockets: the mobile phone. Relying on the security of the phone network alone, however, is risky. Indeed, it is possible for motivated attackers to hijack a phone number to intercept challenge text messages and calls from the identity system.[17] Therefore, it is preferable for the mobile phone to perform some cryptographic operations, ideally using key material protected by the device's firmware or hardware.

Minimal disclosure technologies incorporate the notion of two-factor protection. Indeed, U-Prove tokens can be cryptographically tied to a security device in such a way that the issued token can only be presented if the device collaborates with the user to answer the relying party's cryptographic challenge.[18] Therefore, the second factor provides end-to-end protection of the token, not only at token issuance, both also at presentation (which can happen long after issuance). This extra security protection does not contravene the privacy benefits of using U-Prove.

Our proof of concept implements the U-Prove Device on a mobile phone. We prototyped two approaches to communicate with the phone: one using local communications through the phone's Near Field Communication (NFC) chip, and one using remote calls through push notifications. Both approaches strengthen the security of a transaction.

Here is the flow, as illustrated in Figure 1.

1) **Token issuance**: The User authenticates to the Issuer and obtains U-Prove tokens encoding the public key of her phone (either known to the Issuer, or provided by the User).
2) **Phone invocation**: When presenting the token, the User invokes the phone passing the Verifier's challenge message, either through NFC (red line) or through a remote phone service (blue line). The phone computes its cryptographic response and returns it to the User.
3) **Token presentation**: the User presents the U-Prove token to the Verifier, including the phone's cryptographic response.

[17] See for example http://www.financialexpress.com/news/duo-arrested-for-internet-banking-fraud/1061205

[18] The C# SKD's `UProveCrypto.VirtualDevice` illustrates how to implement a U-Prove Device.

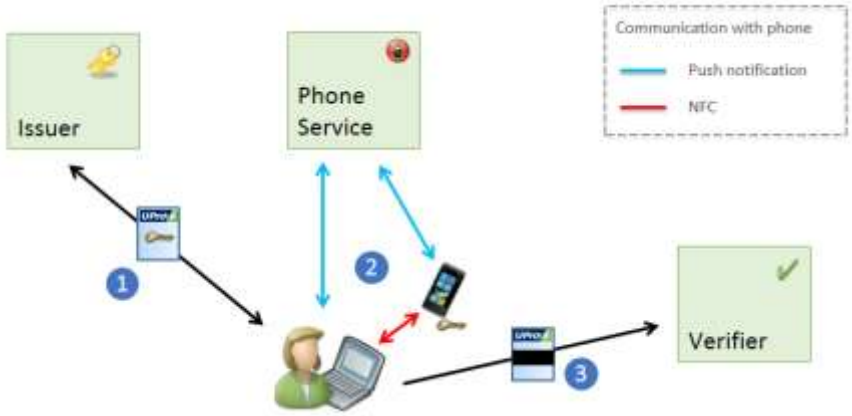

Figure 1: two-factor protection using a mobile phone

If the security device is trusted by the Identity Provider (for example, a tamper-resistant smartcard issued by the same organization), then it can be used to enforce security policies such as validating user attributes, checking revocation status, or encrypting its identity for an escrow auditor. Since mobile phones do not provide sufficient tamper-resistance, we preferred in our prototype to enforce these policies using cryptographic mechanisms.

Relying on a software-only second factor enhances authentication strength, but it is preferable to bind cryptographic keys to a hardware component. One interesting choice is the use of a Trusted Platform Module (TPM) chip. Fortunately, the TPM 2.0 specification[19] allows us to implement a U-Prove Device[20] therefore making it suitable to provide hardware protection of U-Prove tokens.

ID escrow using a verifiable encryption scheme

Auditing is an important feature in identity systems. When something goes wrong, it is important to know what happened and who is responsible. It is trivial to identify misbehaving users when using conventional identity technologies. Indeed, most protocols enforce the use of a unique identifier (for example, the subject field of an X.509's certificate, the user ID in Facebook Connect, or the Subject element of a SAML assertions), and when none is contained in a presented token, forensic analysis can infer which user presented it (by comparing, for example, the logs of the identity provider and the relying party to match cryptographic signatures). Minimal disclosure technologies avoid by design these correlation handles to prevent inescapable tracking by other protocol participants.

How can we, in this case, discover the identity of a fraudulent user if we permit the use of these minimal disclosure technologies? One approach is to build an ID escrow system using a verifiable encryption scheme, as we discuss next.

ID escrow systems have been extensively (and rightfully) criticized in the privacy literature,[21] and indeed great thought must be put into the design of such a system due to the resulting reduction in privacy

[19] https://www.trustedcomputinggroup.org/resources/tpm_library_specification

[20] The TPM 2.0 **TPM2_Commit** and **TPM2_Sign** commands allow us to implement the Device's commitment and response computations, respectively.

[21] The debate around the Clipper chip is the early '90s is a good example.

guarantees. We argue that if accountability requirements are such that fraudulent users *must* be identified, then a carefully designed and auditable ID escrow scheme would *allow* honest users to remain anonymous (rather than relying on conventional identity systems where everyone is fully or easily identified by default).

The idea behind our scheme is simple. The system sets up a trusted Auditor, an entity that will be able to decrypt the identity of a user under certain application-specific circumstances. Each time a user presents a U-Prove token she encrypts a token attribute, encoding her identifier, using the public key of the Auditor. This encryption is provided to the Verifier. If there is abuse, the Verifier forwards this ciphertext to the Auditor to have it decrypted, to learn the user's identifier.

One question becomes evident: how does the Verifier know that the ciphertext is a valid encryption of the user's identifier, and not some junk provided by a malicious user wanting to evade the escrow system? To address this issue, we use a verifiable encryption scheme assuring the Verifier that the encryption is correct, without revealing the encrypted value.

To prevent abuses from the Auditor, a deployment may choose to split its responsibilities among different groups. For example, the Auditor's secret key could be split between a law enforcement agency and a civil liberties group, and be reconstituted and used only when a court order is obtained to lift the anonymity of a particular transaction. These safeguards are application-specific and can be put in place as needed.

Various schemes have been proposed in the cryptographic literature to address this problem. In order to support verifiability, the underlying encryption scheme must be mathematically compatible with the credential scheme. ElGamal is a well-known encryption scheme that works well with the mathematics behind U-Prove, and it has been proposed to implement this feature;[22] extra protocol steps are needed however in order to prove its security. In our case fortunately, we can use a simple ElGamal encryption as the basis of our verifiable encryption scheme, and be assured of its security because it is tied to a U-Prove presentation.[23]

Here is an overview of the scheme illustrated in Figure 2.

1) **Auditor setup**: The Auditor generates its public parameters and secret key, and makes the public parameters available to Users and Verifiers.
2) **Token issuance**: The User authenticates to the Issuer and obtains U-Prove tokens encoding her unique identifier UID.
3) **Token presentation**: the User presents a U-Prove token to a Verifier, including a verifiable encryption of a pseudonym[24] for her unique identifier UID. The Verifier validates the presentation proof and the verifiable encryption by checking that the pseudonym maps to the

[22] See for example: *Practical Verifiable Encryption and Decryption of Discrete Logarithms*. Jan Camenisch and Victor Shoup. http://www.iacr.org/archive/crypto2003/27290126/27290126.pdf

[23] Using only an ElGamal encryption would provide IND-CPA security. Our scheme combined with a U-Prove presentation is conjectured to provide IND-CCA security.

[24] Technically, if x is an encoding of the UID, then the user encrypts g^x for a public base element g. Recovering the discrete log value x from g^x is hard; so as we explain later, either the Issuer remembers the (x, g^x) mapping at issuance, or x must be brute-forced (if the set possible values is small enough).

undisclosed UID encoded in the presented token. The presentation transcript composed of the U-Prove token, presentation proof, and UID ciphertext is stored for auditing purposes.

4) **ID decryption**: In case of abuse, the Verifier sends to the Auditor the presentation transcript along with an application-specific proof of abuse.[25] The Auditor verifies the presentation transcript, decrypts the encrypted pseudonym and returns it to the Verifier.

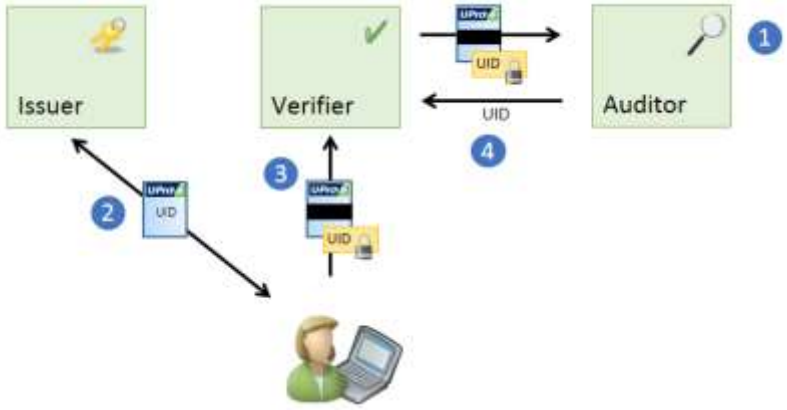

Figure 2: ID Escrow with verifiable encryption

There are two ways to retrieve the UID value from a decrypted pseudonym:

1. The Issuer could keep a mapping between UID values and expected pseudonym values at issuance time, and be queried by Verifiers after an escrow decryption is performed; or
2. The pseudonym value could be brute-forced to figure out the UID value that generated it.

The preferred option is an application choice. If an Issuer can keep the pseudonym value for each user in its database, and if it is accessible after the decryption, then the first option is a possibility. The second option doesn't require the involvement of the Issuer, but is more time consuming. It might however be preferred if the list of possible user identifiers is enumerable and the auditing process is expected to take some time. As an example, assuming the Issuer is a driver's license office, and that the UID is the license driver serial number, then it might be acceptable for a police department to spend a couple of days during an investigation to brute force the UID value given the returned pseudonym.

One interesting option at this point would be to prevent the fraudulent User to present her other tokens; this will become possible using the revocation feature we present in the following section.

For more details about this ID escrow scheme, see the specification available from http://www.microsoft.com/u-prove.

Revocation using a designated-verifier accumulator scheme

User revocation is a mechanism invalidating issued credentials that would otherwise be valid. A credential provider might revoke a credential because the user is not entitled to it anymore (for

[25] The terms defining what constitutes an abuse can be signed by the User as part of the encryption process, therefore preventing a malicious Verifier from claiming that a User abused the service terms by changing them after the fact.

example, has quit his job or changed role) or if a credential was comprised (for example, stolen or hacked).

The most popular revocable credential is the X.509 certificate. Each certificate specifies a serial number that can be put on a Certificate Revocation List (CRL). Before accepting a certificate, relying parties must check that the certificate's serial number does not appear on the current CRL, or query an Online Certificate Status Protocol (OCSP) responder that performs the check. This is simple enough when a unique value is presented with the credential, but as we already discussed, minimal disclosure technologies do not contain, by design, any linkable identifiers.

Various revocation mechanisms are possible when using V1.1 U-Prove tokens. One can limit the validity period of the tokens to make sure the Issuer can periodically validate the user before issuing new tokens. For longer-lived tokens, Verifiers can make an authorization decisions based the token identifier[26] presented from a U-Prove token, refusing to accept tokens for which the identifier has been revoked. For Device-protected tokens a Verifier can rely on the trusted Device (for example, a smart card) to reject user identifiers on the revocation list passed to the Device.[27] However, when multiple unlinkable software-only long-lived tokens are to be issued, which is the case in many identity scenarios, we need another mechanism to effectively revoke users.

Multiple revocation schemes for minimal disclosure technologies have been proposed over the years. The simplest consists of performing multiple negation proofs: disclosing a token attribute is equivalent to proving that an attribute x is equal to a certain value v $(x = v)$; it is also possible to mathematically prove that an attribute value is NOT equal to a value $(x \neq v)$. Assuming that the revocation list contains the identifiers of all revoked users $(v_1, v_2, \dots, v_n)$, then a non-revocation proof consists of proving that $(x \neq v_1, x \neq v_2, \dots, x \neq v_n)$. The problem with this approach is that the proof's computation time and size is linear to the size of the revocation list. Clever approaches have been proposed resulting in schemes that have square root[28] and logarithmic[29] complexities in the size of the revocation list.

Dynamic accumulators[30] use a different approach in which the revoked values are accumulated into a single aggregate value by a party called the Revocation Authority. The benefit of these schemes is that they allow users to create non-revocation proof in constant size and time, once some accumulator pre-computations are made. These pre-computed values must be updated anytime that revocation list changes, which makes the scheme interesting if this happens in well-defined intervals.

Most dynamic accumulator schemes use a type of algebraic constructions called bilinear pairings. Although pairings are popular in recent cryptographic research, they are not yet used in practice due to their maturity level and implementation complexity.

---

[26] The token identifier is a unique value defined as the hash of the token's public key and signature.

[27] Messages passed to the Device are signed by the User and validated by the Verifier, preventing the User from removing her identifier from it.

[28] *A practical system for globally revoking the unlinkable pseudonyms of unknown users*. Stefan Brands, Liesje Demuynck, and Bart De Decker. http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW472.pdf

[29] *Zero-knowledge Argument for Polynomial Evaluation with Application to Blacklists*. Stephanie Bayer and Jens Groth. http://www0.cs.ucl.ac.uk/staff/J.Groth/PolynomialZK.pdf

[30] *Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials*. Jan Camenisch and Anna Lysyanskaya. http://www.zurich.ibm.com/security/publications/2002/camlys02.pdf

In systems where the Revocation Authority and the Verifiers share a key (hence the term "designated" verifier), then we can design an accumulator scheme that uses a conventional finite field or elliptic curve construction (like the ones used in DSA and ECDSA, and in U-Prove).

Here are the details of the scheme illustrated in Figure 3:

1) **Revocation Authority setup**: The Revocation Authority generates its public parameters and secret key, and makes the public parameters available to Users.
2) **Token issuance**: The User authenticates to the Issuer and obtains U-Prove tokens encoding her unique identifier UID.
3) **Revocation list management**: Periodically, the Revocation Authority updates the revocation list, and the User obtains non-revocation witnesses from the Revocation Authority or computes them using the updated list values.
4) **Token presentation**: The User presents a U-Prove token to the Verifier, including a non-revocation proof (computed using the non-revocation witnesses). The Verifier validates the presentation proof.
5) **Revocation verification**: The Verifier sends the non-revocation proof to the Revocation Authority that verifies that the undisclosed UID does not appear on the current revocation list. [31]

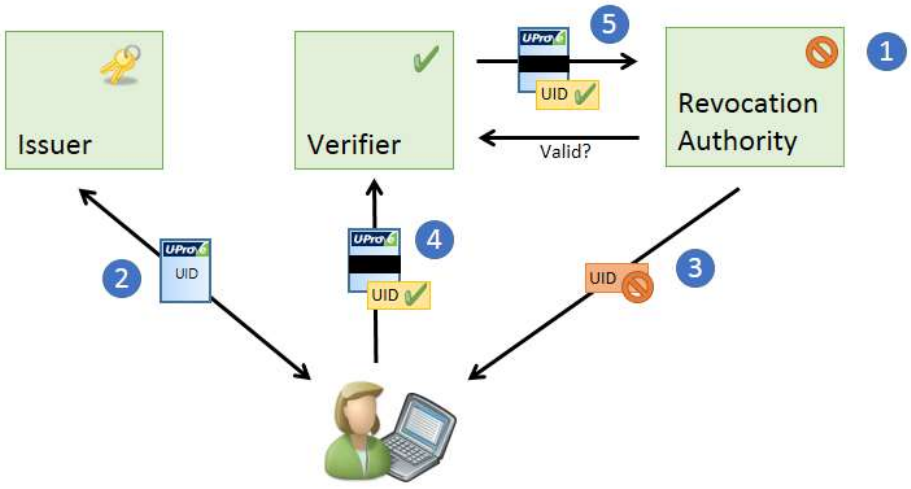

Figure 3: Revocation with Accumulator and Designated-Verifier

For more details about this revocation scheme, see the specification available from http://www.microsoft.com/u-prove.

Proof of concept implementation

We implemented a proof of concept illustrating all these components working together. The scenario is as follows. Alice visits a Privacy Marketplace where she can purchase music, games, apps, etc. The Privacy Marketplace protects users' privacy and do not track their every purchase. Instead of charging Alice's credit card every time a purchase is made, Alice purchases in advance a number of anonymous

[31] Much like an OCSP call when using X.509 certificates.

tokens that can be redeemed at various merchants in exchange of an online item. Neither the Marketplace nor the merchants, even in collusion, can link a token issuance with a token redemption.

To strengthen security, Alice must confirm with her mobile phone Marketplace purchases made with her laptop. The tokens are bound to a key stored on the phone, and the phone is needed at purchase time to answer the merchant's cryptographic challenge.

Each token should only be used once, and deleted afterwards. To prevent fraudulent users from hacking the client application and cloning tokens, the Marketplace encodes in each one the user's account identifier as an attribute. When making a purchase, Alice hides her identifier, but creates a verifiable encryption of it for the system's Auditor. When the Marketplace validates a token, it keeps a list of all presented token transcripts indexed by token identifiers.[32] If there is a match between a newly received token and an indexed one, the Marketplace concludes that the token was presented twice, and therefore that the user misbehaved.[33] The current and the stored transcripts are sent to the Auditor which decrypts the identity of the user, and returns it to the Marketplace which in turn adds it to its revocation list. From now on, Alice will not be able to use any of her tokens, since the revoked identifier is encoded in all her issued tokens.

In this prototype, the Privacy Marketplace service implements the U-Prove Issuer and Revocation Authority roles, user application implements the Prover role, and the phone implements the Device role.

Here are the steps illustrated in Figure 4.

1) The User logs in her Marketplace account using her username/password and purchases U-Prove tokens encoding her UID ("alice"). The tokens are bound to the User's phone associated with her account.

2) The User periodically obtains non-revocation witnesses from the Revocation Authority.

3) The User obtains an item from a merchant service by presenting a token, encrypting her UID for the Auditor, providing a non-revocation proof of her UID and a cryptographic response from her phone.

4) The merchant service calls back the Marketplace for token validation. The Marketplace stores the presentation transcript.

5) In case of abuse, Marketplace sends the presentation transcripts to the Auditor to learn the user's UID and adds it to the revocation list.

[32] A cryptographic hash of the token's public key and signature.

[33] An application should have fault-tolerant mechanisms to detect and prevent erroneous (vs. malicious) behaviors.

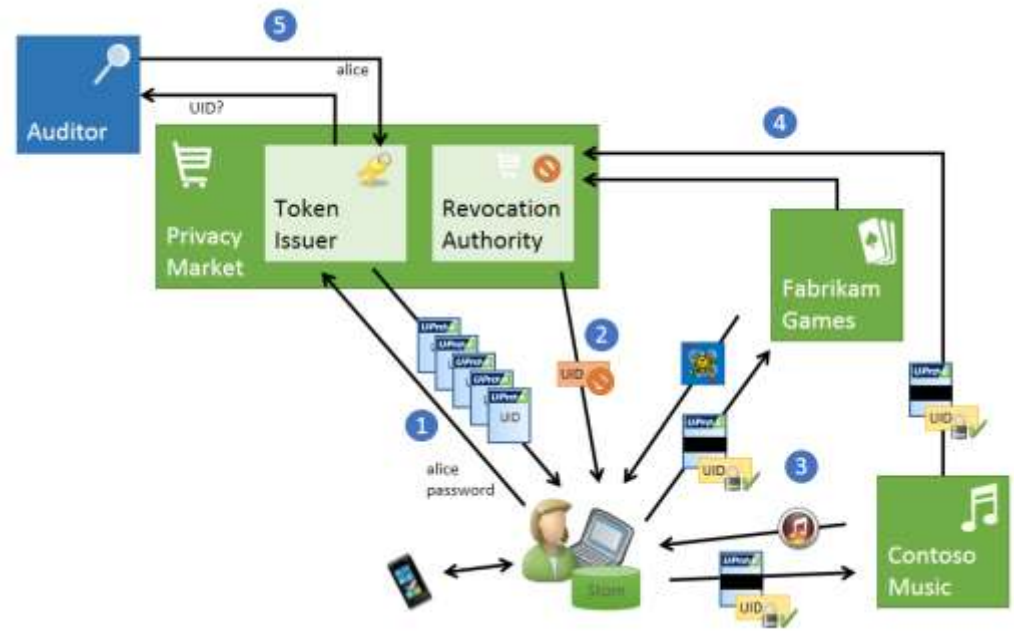

Figure 4: Proof of concept architecture

This prototype was implemented using a Windows RT client application for the User, a Windows Phone 8 application for the Device, and back-end Web Services for the Marketplace and merchant services. All of them use the U-Prove C# SDK. We configured the Issuer to use the P-256 elliptic curve to improve performance and reduce communication complexity.

Conclusion

We described strong accountability mechanisms that can be used in conjunctions with strong privacy technologies, demonstrating that *both* security and privacy requirements can be achieved in identity systems.

We encourage you to learn more about the U-Prove technology and the new capabilities by visiting http://www.microsoft.com/u-prove, and to incorporate these capabilities into systems you are designing to improve their security and privacy protections.

Questions and feedbacks can be sent to uprove@microsoft.com.

Acknowledgments

This paper highlights the work of my esteemed XCG colleagues Greg Zaverucha, Lan Nguyen, and Melissa Chase.