Copy-Resistant Credentials with Minimum Information Disclosure

David Bauer and *Douglas Blough*

Georgia Institute of Technology

Public-key based certificates provide a standard way to prove one's identity, as certified by some certificate authority (CA). But standard certificates provide a binary identification: either the whole identity of the subject is known, or nothing is known. By using a Merkle hash tree structure, it is possible for a single certificate to certify many separate claims or attributes, each of which may be proved independently, without revealing the others. Additionally, trees from multiple sources can be combined together by modifying the tree structure slightly. This allows claims by different authorities, such as an employer or professional organization, to be combined under a single tree, without the CA needing to know (let alone verify) all of the claims.

1. Introduction

Personal information is increasingly used to establish identity and authorize transactions in the digital world. At the same time, identity theft and fraud, based on unauthorized disclosure and misuse of personal information, are rampant [7], and individuals are increasingly concerned about providing personal information to every digital entity with which they establish a relationship. The research described in this technical report is based on several key principles of identity management:

- first and foremost, users should have the maximum control possible over what personal information of theirs is disclosed in any given on-line interaction,
- next, more reliance can be placed on personal information that is verified by trusted third parties than in self-reported information, and
- last, if verified personal information is to be used, mechanisms to prevent that information from being copied and misused by unauthorized parties are essential.

We assume an architecture in which there are identity providers that verify users' personal information and supply *credentials* that the users can give to service [1][2]. Credentials are a common mechanism for verifying personal information in everyday life. Most people carry multiple physical credentials with them, from drivers' licenses to insurance cards to credit cards. A credential describes some set of attributes about the holder. For the obvious example, a driver's license states that the holder is licensed to drive a vehicle in the licensing state. However, due to drivers' licenses being all but universal, they are used as a general credential. As such, driver's licenses often include unnecessary information, such as the holder's date of birth, address, height, organ-donor status, and social-security number. Electronic credentials can be simple, like a user-name and password, or more complex, like a public key infrastructure (PKI) certificate. A PKI certificate is an electronic document that holds an identity and a public key, and that is signed by a certificate authority (called the issuer). The user holds the associated private key to prove that they are the legitimate holder of the certificate. The user-name and password combination is the most widely used scheme, because of its simplicity. However, this scheme also provides no direct information about the user. A user-name must be attached to a previously made account, and some other form of credential must be used to tie an identity to the account. Such accounts are very seldom shared between different domains, leading users to accumulate many different accounts, often with different user-names and passwords. PKI certificates for users are less common, but can solve the problem of needing to keep track of many different user-names and passwords.

Minimum information disclosure in any given interaction is desirable from a user's perspective, and may even be necessary for a given technology to be widely adopted [3]. Clearly, if a user wants to release the minimum amount of personal information on a given interaction, this rules out a single credential approach, in which each user maintains one credential containing all of their personal information and uses that credential for every interaction. As in [3], we say that a user makes a *claim* about herself when she gives information about one or more personal attributes to a digital entity. Due to the wide variety of personal information that is used in digital interactions, the number of different possible claims is extremely large. The problem to be solved is, therefore, to provide an efficient and reliable mechanism that allows users to assert arbitrary (or at least a large number of) verifiable claims over a sequence of interactions with different digital entities.

As an example of minimum information disclosure, consider the problem of verifying that a user is at least 18 years of age. Clearly, verifying the user's date of birth is sufficient but not necessary, and would reveal a very sensitive piece of personal information that could assist an identity thief in masquerading as that user. We refer to the claim that a user "is at least 18 years of age" as a micro-claim, with the (macro-)claim in this case being "the user's date of birth is xx/xx/xx". Many different micro-claims can be derived from a single claim, e.g. the user is "at least 18", "at least 21", "at least 35", "at least 65", etc. One possible approach could be to maintain credentials for a relatively small number of claims and use those to generate dynamically micro-claim credentials as needed for a given interaction. However, how to automatically generate a large variety of micro-claims from a given set of claims is an open problem, not to mention how the micro-claims can be verified without revealing the information in the verifiable claims from which they are derived.

Instead, we adopt an approach where a large set of micro-claims is enumerated statically, and updated dynamically as needed. Instead of generating and maintaining a single credential per micro-claim, which is extremely inefficient from the standpoints of storage space, bandwidth, and computation time, we propose a method wherein a single credential can be maintained that allows the user to dynamically specific an arbitrary subset of micro-claims for a given interaction without revealing the other micro-claims. We also include in this credential, and the protocols that use it, mechanisms to make it very difficult for an attacker to make a copy of the credential and use it to masquerade as the user. As an added benefit, the credential allows information that is verified by different identity providers to be combined in one structure. This allows users to spread out their personal information across different identity providers, thereby lowering their risk when one of their identity providers experiences a security breach. The details of this new credential mechanism and its associated protocols are provided in the remainder of this report.

2. Expected Scenario

We consider a scenario with three types of parties: identity providers, service providers, and users. An identity provider is an entity in a position to make authoritative statements about a user. An identity provider can be a third party certificate authority, a government office, an employer, a professional organization, or more. A service provider is an entity that wants or needs to check the identity of users. A service provider can be any organization, government office, or individual with an on-line presence. The user is the holder of the credential, and is always an individual person in our scenario.

3. Background

The design of our electronic credential was driven by several requirements. First, the credential must not simply hold a single identity. Identity is a complex matter, and often a name or serial number is not what is important in one's identity. Second, the user of the credential -- the one whose identity is being proven -- must have as much control over the process as possible. Considering the first requirement, the user should be able to select which attributes of their identity are released to a particular entity. Third, the credential must have some form of copy protection. Using the credential should not place it at risk of being copied. Fourth, neither the user nor the service provider should have to contact the identity provider to verify the credential. Fifth, the credential must be memory and computationally efficient to store and use.

4. Design

4.1 Merkle Hash Trees

The proposed credential is based upon Merkle hash trees [4] and standard public-key infrastructure (PKI) certificates. A Merkle hash tree is essentially a binary tree where each internal node holds the hash of the concatenated values of its two children nodes. The leaf nodes hold the data of interest. In this way, a large number of separate data can be tied to a single hash value. In addition, by storing the internal node values (or a subset thereof), it is possible to verify that any of the leaf nodes is part of the tree without revealing any of the other data. Merkle first introduced this structure as a way to efficiently handle a large number of Lamport one-time signatures. It has since been adapted for uses such as the large-scale time-stamping of documents [5] and tracking data in peer-to-peer networks [6].

A basic Merkle tree is shown in Figure 1, below. Figure 2 shows the same tree with the nodes and claims labeled for easy reference. Consider verifying claim A in the tree. Starting at the claim and going up the tree, node 4 contains the hash of the claim. Node 3 contains the hash of the concatenation of nodes 4 and 5. Node 2 contains the hash of the concatenation of nodes 3 and 6. And, finally, node 1, the root of the tree, contains the hash of the concatenation of nodes 2 and 9. Therefore, verifying claim A requires the values of nodes 5, 6, and 9, as shown in Equation 1, below. A similar verification path can be made for any of the claims. For example, Equation 2 shows the path for verifying claim F, which requires the values of nodes 2, 11, and 13. As per the nature of the binary tree, the number of nodes and hashes needed for a verification scales with the log of the number of claims.

Root value = H(H(H(H(A), 5), 6) 9) Equation 1

Root value = H(2, H(H(11, H(F)), 13) Equation 2

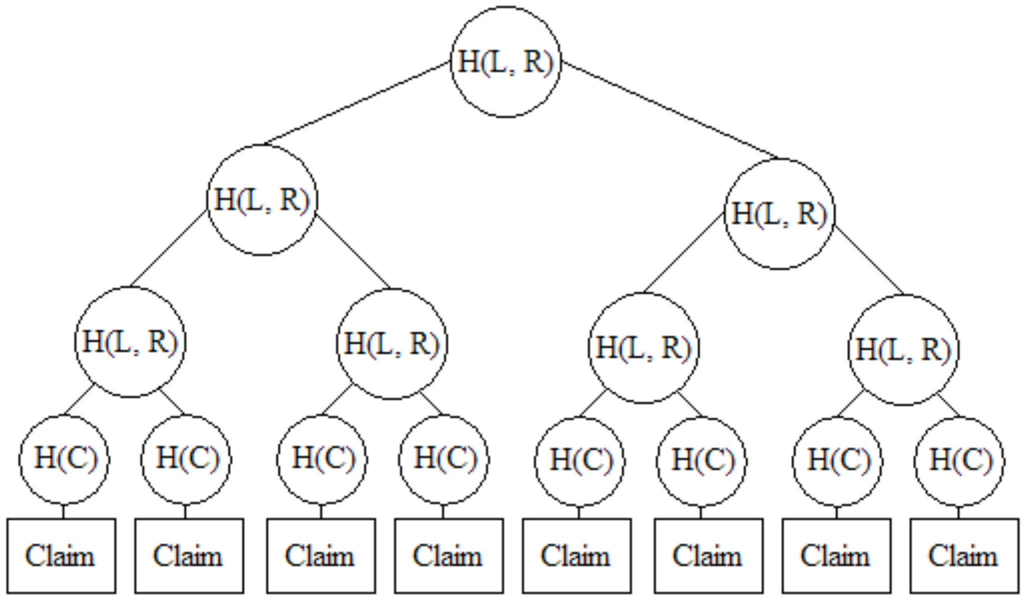

Figure 1. Merkle hash tree with leaf nodes holding hashes of claims.

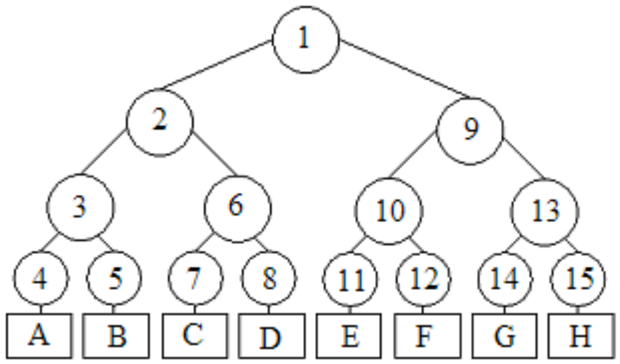

Figure 2. Merkle hash tree with labeled nodes.

4.2 Credential Overview

The credential consists of two parts: a public part and a private part. The public part of the credential is a certificate. The certificate holds information about the issuer, the certification chain for the issuer, the type of certificate, the date range over which the certificate is valid, the user's public key, and a signature of the root node of a Merkle hash tree. The certificate should not in general hold any data about the user directly, even data as common as a name. As per standard operation, the certificate will be signed by some certificate authority, which is an identity provider in this system. The private part of the credential consists of a private key and a Merkle hash tree whereby all of the leaf nodes are attributes of, or micro-claims about, the identity of the user, who is the credential holder. The Merkle tree structure allows the credential holder to prove any subset of the claims in the tree, with only the single signature on the certificate.

As an additional improvement, a slight modification to the structure will allow the use of a single credential containing claims from a variety of identity providers, without requiring one identity provider to verify all of the claims. Consider again the Merkle tree of claims, but with subtrees coming from different identity providers. For example, one subtree could contain claims certified by a government registry, while another subtree could contain claims certified by an employer. In the basic structure, the identity provider must either see all of the claims or trust the providers of all of the subtrees that they only contain claims relevant to their topic. Neither solution is ideal. To provide a third option, consider adding an optional third branch to some internal nodes of the full tree. (These nodes correspond to root nodes of the subtrees.) The third branch contains a certificate for all of the claims in the subtree rooted in the parent node. When verifying a claim which is in such a subtree, the form is different -- three hashes are concatenated in a node instead of two -- so the verifier knows that this claim is under a different certificate than the credential as a whole. For example, in Figure 3, below, the left hand subtree contains claims certified by the overall identity provider. The overall identity provider is referred to as the certificate authority (CA). The right hand subtree contains claims certified by some other party, which do not have to be verified by the top level CA.

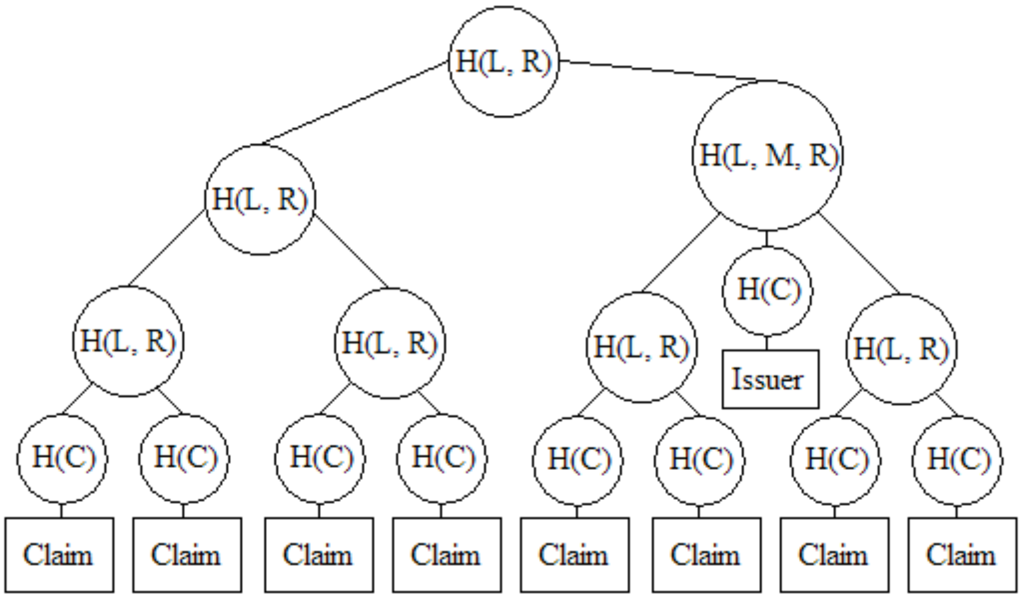

Figure 3. Modified Merkle hash tree with labeled subtree.

4.2 Protocols for the Credential

Creating a credential is both conceptually and computationally easy. In the case where there is a single identity provider for the credential, there are roughly four steps required:

1. Agree on a list of claims.
2. Generate the hash tree for the claims.

3. Verify that the user possesses the private key.
4. Produce and sign the public certificate.

First, the user and identity provider agree on a list of claims. The logistics involved in an identity provider verifying the user claims may be rather complicated, but are beyond the scope of this paper. Second, either the user or the identity provider generate the hash tree for the claims. Random padding must be added to the claims before they are hashed, and such padding can conceivably be used to leak information. If there is a concern, then a cooperative protocol can be used to generate the random padding. With a single identity provider, the tree will always be balanced, bounding the number of interior nodes to the number of claims. The number of hashes needed to generate the tree is therefore bounded to twice the number of claims -- generally a negligible amount of computation time. Third, the identity provider must verify that the user holds the private key matching the public key of the credential. The user can reuse an already generated key pair or generate a new key pair for the credential. The identity provider does not need to ever know the private key. Finally, the identity provider creates and signs the public certificate for the credential.

Creating a credential with claims from multiple identity providers is very similar to creating a credential from a single identity provider. One identity provider will be the final one, referred to as the certificate authority. The user creates credentials with all of the identity providers except for the final one, by the procedure described above. Then, the user creates a credential with the CA. For this final credential, the hash trees from the other credentials are incorporated into the final hash tree. The root nodes for the incorporated subtrees will have a third branch added, containing the certificates from the original identity provider of that subtree, as shown in Figure 3, above. (The root hash values in the subtree certificates will no longer match, but that doesn't matter.) The CA verifies the values of just the top two sub-nodes of each subtree, as well as the public part of the sub-credentials and the associated public/private key pairs. The public/private key pairs may be the same as or different from each other and the top level key pair. All that matters is that the user holds the corresponding private keys at the time that the credential is assembled. The CA does not need to see the claims (other than the public key), or even any more of the subtree structure.

The protocol for using the credential follows conventional PKI certificate usage. The user connects to a service provider, either over a wide or local area network, and requests some service, sending the public part of the credential. The service provider requests the appropriate identity attributes from the user. The user provides the claims which match the requested attributes and the intermediate node values and path information necessary for the service provider to verify each of the claims. (If any of the claims are in a subtree certified by a different identity provider, the accompanying certificates must be included with the set of claims.) The service provider verifies that the claims are in the hash tree specified (via the root hash) in the public part of the credential. The service provider also verifies the signature on the public part of the credential and on any subtrees from different identity providers. In order to verify that the user is the holder of the credential, the service provider also verifies that the user possesses the private key which matches the public key claimed by the credential. This can be done by standard methods, such as challenge/response or as part of a secret key agreement operation.

4.3 Copy Resistance

The credential is strongly copy resistant under normal operations, due to the incorporated public/private key pair. Whenever the credential is used, the private key is used to prove that the user is the authorized credential holder. Since the private key never leaves the user's machine, it can't be copied by the service provider or any third party listening to the exchange. (The private key can of course be copied if the user's machine is compromised, but that is beyond the scope of this paper.)

This copy resistance property extends to preventing standard man-in-the-middle attacks. For example, consider how a phishing site handles conventional one-time passwords or other two-factor authentication methods. A user goes to login to what they believe is the legitimate site of their bank, broker, or other service provider, but which is really a phishing site. The user enters their user-name and password and then, either in the same step or in a second step, enters a one-time password from a sheet of paper or electronic device. The phishing site simply passes all the information from the user on to the site being spoofed, and returns the responses from the legitimate site to the user. After the user has successfully logged into the legitimate site, through the phishing site, then the phishing site is logged in as the user and can perform whatever actions it wants.

The standard protection against this type of man-in-the-middle attack is the use of server-side certificates with SSL/TLS. When the user initiates a secure connection to a service provider, the service provider replies with a PKI certificate. The user's client (web browser) usually handles checking the validity of the certificate automatically, interrupting the user only when a problem is detected and simply showing a non-intrusive indicator when the secure connection is setup without problems. Phishing sites have used a variety of techniques to get around server-side certificates, including simply not using SSL/TLS, making the browser look like it is using SSL/TLS when it is not, and obtaining valid certificates for different domains, which can be mistaken for the legitimate domain by an unwary user. All of these tricks are possible because of the disconnect between the authentication system (SSL/TLS), and the thing being authenticated (the service provider). The certificates authenticate the domains used, instead of the actual service providers.

In contrast to the server-side certificates, our credentialing scheme intimately ties the authentication system (the private/public key pair and the certificate as a whole) to what is being authenticated (the credential as a whole). Consider again the phishing site, but with the credential in place of the user-name, password, and one-time password. When the user connects to the phishing site, the phishing site can relay requests to the legitimate service provider and replies back to the user as before. However, when the service provider sets up a secure connection, it will use the user's certificate to setup the session. The phishing site can substitute its own certificate, but then it is no longer impersonating the user. Alternatively, the phishing site can pass on the user's certificate, but then the phishing site loses control of the session. The service provider can setup an authenticated tunnel to the user, so that the phishing site can no longer modify the traffic without being detected.

5. Conclusion

Privacy is important both as a protective principle and as a security measure. Identity theft is a serious and widespread crime. The Federal Trade Commission reports that over a quarter of a million identity theft complaints were received in 2005, in addition to over 430,000 other fraud complaints. Internet-related complaints accounted for almost half of those [7]. Protecting personal information is vital to reducing identity theft. Limiting information disclosure does not require that accesses to service providers be anonymized. A user is assumed to always present the same credential to a service provider, allowing for tracking of suspicious behavior. Depending on the nature of the service provider, accumulated user information may be sent to the identity providers for auditing.

References

[1] Dick Hardt, "Identity 2.0," presentation at OSCON 2005, 2005 Aug 4, [Online document] [2006 Nov 27], Video available HTTP: http://www.identity20.com/media/OSCON2005/

[2] "Microsoft's Vision for an Identity Metasystem," [Online document], 2005 May, [2006 Nov 22], Available HTTP: http://msdn2.microsoft.com/en-us/library/ms996422.aspx

[3] K. Cameron, “The Laws of Identity,” [Online document], 2005 May 12, [2006 Nov 22], Available HTTP: http://www.identityblog.com/?page_id=354

[4] R. Merkle, "A Certified Digital Signature", Advances in Cryptology--Crypto'89, 1989.

[5] D. Bayer, S. Haber, and W.S. Stornetta, “Improving the Efficiency and Reliability of Digital Time-Stamping”, Proceedings Sequences II: Methods in Communication, Security, and Computer Science, Springer-Verlag, pp. 329-334, 1993.

[6] J Cates, “Robust and Efficient Data Management for a Distributed Hash Table”, Master’s Thesis,

Massachusetts Institute of Technology, May 2003.

[7] "Consumer Fraud and Identity Theft Complaint Data," [Online document], 2006 Jan 25, [2006 Nov 17], Available HTTP: http://www.consumer.gov/sentinel/trends.htm