Blind Public Key

Scalable disclosure-free credentials for e-business that model real trust relations

Andrew Yeomans, VP Global Information Security, Dresdner Kleinwort Wasserstein

Abstract

Current password systems are critically flawed due to attacks from keyloggers. Also an attack that compromises a credential database will allow the shared secrets to be used on other accounts. PKI systems don't correctly model real trust relationships, hence the slow adoption. The presentation shows how public key technologies can overcome the issues, with a real world example of Internet ecommerce. Hardware token designs are considered to provide the mutual non-replayable authentication, with a migration from existing password techniques.

Introduction

This paper proposes a revised identity credential scheme that can be used both for internet and for workstation authentication.

Issues with existing credential systems

The majority of user authentication schemes today still use userid and password. This is a technology that should have been superseded a long time ago - software and hardware key-loggers easily capture this data and allow others to use the same credentials. Massive fraud attempts have been seen using such key-loggers. Network sniffers too can also capture userids and passwords. Hence any proposed system should be **non-replayable**, thus preventing any misuse of captured passwords.

Two-factor authentication is often proposed to overcome key-loggers. Strictly speaking this is a misunderstanding, since many commercial two-factor authentication systems are also non-replayable. But that is not inherently true – for example a dual password system is two-factor, but completely vulnerable to key-loggers. Two-factor and other strong authentication schemes raise another issue, that the authentication should be **proportionate** to the use. Just because it's possible to use strong authentication does not make it desirable. It is possible for an on-line newspaper to require biometric authentication before it can be accessed – for example Bloomberg has added finger-print readers to its keyboards. But it detracts from the usability, thereby helping their competition, and may additionally fall foul of legislation on personal data.

Phishing is an increasingly common fraud technique, when a fake web site is created to mimic a real one. The victim is lured to the fake site, either by URL links in an email pointing to the fake site, or by having a web site name confusingly close to the real one to pick up typing errors. And the real site will almost certainly be held responsible for the confusion, despite it being out of their immediate control. The solution is well known – namely **mutual authentication**. The problem is that although current systems such as SSL web sites sites technically allow the customer to validate the web server, in practise the user actions to check this (click on a padlock, read obscure messages) means this is rarely done; and in any case it is difficult to get any more guarantee than the web site was signed by some Certificate Authority (CA). CAs have been seen to make mistakes; but more importantly they are not guarantors of trustworthiness and may well honestly register a legitimate company with a name which was intended to defraud. So the mutual authentication must also be **automatic**.

Man-in-the-middle attacks are also expected to become more widespread, as communications methods move away from relatively well protected cables run by regulated service providers, towards ad-hoc wireless communication systems. It's extremely easy to create a spoof wireless access point that pretends to be a well-known wireless ISP, and to capture any account details being passed. An even bigger threat comes from Trojan Horse software that will wait for a user to connect to a finance site and then surreptitiously conduct transactions. The **mutual authentication** requirement above helps prevent some man-in-the-middle attacks, but an even stronger precaution is to use **end-to-end** authentication – ideally from the human individual at one end to the human or corporate individual at the other.

Traditional PKI systems use a Certificate Revocation List (CRL) as a black-list. Although this can work in smaller-scale systems, it depends on all key losses to be reported, and also depends on there being far fewer revoked certificates than accepted ones. There have been reports that the US Department of Defense smartcard system has CRLs of 40 Mbytes. That size of CRL adds a production problem, distribution problem and a maintenance problem. In addition, this CRL approach runs counter to the normal security philosophy of "default deny". So a requirement is for a **scalable** technology using a white-list approach.

There should be **no re-used secrets** in the whole system otherwise compromise of a single secret will lead to many other secrets being immediately discoverable. The obvious example is the common warning not to re-use passwords or credit-card PINs. Users don't always remember that attackers might be insiders; the staff of an outsourced merchant system probably have access to password and other identification information that could be used on other accounts. Federated identity systems also run this potential risk, where the risk of compromise increases with the number of organisations in the federation. So we need a **disclosure-proof** credential system.

Identity transactions in the electronic world need to model identity transactions in the real world. Over time, humanity has evolved ways of doing business in an untrustworthy world. Systems which ignore this are likely to hit serious problems. The traditional Public Key Infrastructure approach is a good example; the original plans were to have a few trustworthy CAs who would be granted the signature of approval by the government, with some United Nations "root CA" body signing the government keys. You don't need to know much history or politics to see that not all governments are trustworthy, or are trusted by all their citizens. So part of this requirement could be stated as **not devolving trust** to other authorities, in other words giving people the power to make **responsible decisions themselves**. This does not mean that authorities have no place in the trust decision process; governments or trade bodies can give excellent advice on fraudulent companies.

Credentials must also be highly **mobile** and capable of being used with a wide range of devices of varying capabilities. In practise, several credentials may be used by the same individual in different circumstances; so the concept of a one-to-one relationship between individual and credential should be discarded.

How do we get there from here? Very occasionally it is possible to have a "big bang" approach to introduce a new system, but usually this is only possible within an area under control of a single authority. Even medium size companies have problems replacing existing systems, due to the need for agreement from so many parts of the organisation. Excellent technologies have great difficulty displacing the incumbent one due to this natural inertia. So another requirement is for the technology to be **incrementally achievable**.

While writing this paper, I discovered a related paper by Kim Cameron entitled "7 laws of identity". These will also be considered as requirements for credentials for trusted electronic commerce.

Proposal

This proposal uses public key techniques to offer an excellent solution to many of the issues listed above. Each of the individual parts of the solution have been used elsewhere; the novelty is in the combination.

Taking the example of web site authentication, the proposed scheme uses unmodified SSL network protocols to provide the authentication. Both client and server X.509 certificates are used, in order to permit mutual authentication. However, contrary to traditional PKI usage, the client certificate contains no identification information or authentication information; it is used solely as an anonymous credential. But surely this loses the power of PKI to provide an authority to vouch for the user? Precisely! As the requirements above have shown, that trust model does not truly reflect the real commercial world, and has led to adoption failures in PKI.

In the real world, business transactions often start with a complex validation process, which may include setting up legal contracts, taking out credit references, and examining multiple credentials to reduce risk of impersonation. And PKI cannot do any of that. Once the transaction partners have concluded their validation process, they are really only concerned that impersonation cannot take place in the future, and they are still dealing with the person or organisation that the contract was with.

So an anonymous or "blind" credential (you cannot see anything hidden within it) is perfectly adequate for subsequent transactions, once the validation process has completed and the credential registered to the two parties. A truly anonymous credential can also be passed between organisations, without any privacy-infringing information leakage in itself.

Companies will still require additional information gathered by that validation process, and which may include name, address, email address, referee, credit card details, and authorisation information. It's expected this would be stored in a database, just as in current schemes, subject to the same data protection and access regulations. But none of that information is held in the credential itself. Companies may still pass on customer details to others; but for credentials, all they can pass is the anonymous public key certificate.

Similarly, the customers can also pass company details to others, as recommendations for or against. And such recommendations can be built into a "web of trust" if desired, to help the users in their decisions. But is not an essential part of the process.

A major advantage of the use of public keys is that the credential information is not secret at all, so that if a public key credential database is stolen, the credentials themselves need not be replaced as the secret keys have not been compromised. Additionally, public key authentication is inherently non-replayable (with a good implementation), as different challenges will be generated each time, and compared with the encrypted response.

Client certificates

The SSL protocol requires normal X.509 certificates to be used, which require the public key to be signed by a CA. So how do we get anonymous credentials without using a third-party CA? The easiest way is to make the process that generates the key-pair also sign the certificate. But not as a self-signed certificate, instead sign it with a well-known key pair. That key pair would be distributed in the signing software, placed on web sites and widely publicised so anyone can sign certificates with it.

A well-known certificate is used since the SSL/TLS client certificate request in the protocol requires the server to submit a list of CAs that are acceptable. By adding the well-known certificate, the anonymous client certificate will be selected by the browser. So no changes are

required in the protocols or software, simply an additional certificate adding to the server-supplied list of root certificates.

So, in summary, the client generates a random key pair, assumed to be statistically unique, then signs it with a well-known CA with published private key, to create the public certificate. One minor enhancement is to generate a pseudo-random sequence number, possibly as a hash of the public key, since some SSL implementations have problems if duplicate serial numbers are seen.

That public certificate is then registered with the counterparty, following appropriate business validation processes. They may involve out-of-band signals (phone call validation), one-time passwords (credit card PINs, personal visits – whatever is appropriate for that business. Once one certificate has been registered, other certificates may be added using the validated certificate as a credential. In this way multiple certificate stores may be provided, such as personal computer, mobile phone, hardware token, with different certificates so the loss of one certificate does not compromise the others. So "backup" certificate stores can be created, without having to share the private keys.

Server certificates

At the server end of the SSL connection, the server supplies the well-known public certificate, requests the client certificate, and then checks that certificate against its registered customer list. Only valid customers will be in the database – this is a "white-list" approach. A database entry may contain several certificates for the same customer, and also contains whatever authorisation permissions that have been granted that customer. There is never any need to check certificate revocation lists, as unrecognised certificates will be stopped.

In just the same way as the server check, the client will also perform a similar check, to ensure that the server's certificate is on the list of approved counterparties for business. That list is added to whenever a new business relationship is validated. So the whole business validation and registration process is symmetrical; the client provides a certificate to the server for future checking, and the server provides a certificate to the client.

That means that if a phishing attempt is made, the phishing site will not be recognised, so the client will be warned of the impersonation attempt. Similarly a man-in-the-middle attack will also be detected, as the attacker does not have either private key.

The client is in control of their privacy. If they wish, for convenience, they can re-use the same certificate for several different counterparties. This is similar to the use of an identical "secret" password to access multiple web sites requiring registration. Or they can use different certificates for each site, to prevent the authentication process leaking identity information.

Other uses

The above sections have described web-based authentication. Exactly the same approach can be used for workstation login, especially if hardware-based tokens are used rather than software credential stores in a browser.

SSH certificates can be used in the same way; current ssh implementations use different key formats, but the same principles are in use. SSH has had the impersonation detection for a long time. It helps usability if similar user dialogues can be shown, and there is great scope for such rationalisation.

How to phase this in

Most of the details require no changes to the technology, simply minor changes to the processes of registration, and configuration to accept client certificates as credentials. The most significant changes are at the client end.

A first stage is to provide a browser extension to perform the server certificate checking. This is highly desirable in any case, even without the full blind public key approach. An automatic warning that web sites have a different or unrecognised certificate would be a great improvement to defend against phishing. (Mozilla Firefox has a "Petnames" extension providing a similar function.) If integrated into a browser authentication store, this could be made simple to use and control, also allowing a gradual migration from userid/password authentication to client certificates.

So the authentication store would hold details of the web site name, its server certificate (if any) for validating; the client certificate for that transaction, or older userid and password; and also the level of protection required. The protection level is to provide usability and proportionality – for example newpaper sites may be automatically logged onto without asking the user; home banking sites might require biometric or PIN input; and intermediate risk sites might just require an "OK" to be clicked.

That system will work well as a replacement for a password store, with a natural upgrade from passwords to client certificates. Blind public key certificates are only one of the supported technologies, avoiding migration issues.

Hardware token

Any use of a general-purpose computer has risk of Trojan Horse software being installed which could compromise the authentication store. So it would be preferable to migrate that onto a dedicated hardware device, simple enough that any firmware could be verified clean and free from malicious code. This is likely to be essential for workstation logon, since there is no good way of unlocking a software authentication store without yet another authentication device. But a separate hardware device could do both.

Today's technology is certainly capable of creating such a device, and with good marketing generating a large market, could sell it for only a few dollars. One problem is the lack of a good interface standard common to all PCs, which meets all the user needs. USB, smartcard, wireless or infra-red all offer reasonable choices, but cannot be guaranteed to be found on all PCs. This is one reason why the ability to make backup copies of the authentication store is invaluable, as it allows different devices to be used as appropriate, until a smaller number of preferred standards emerge.

A second problem is the need to install device drivers – which again restrict the usability of devices. Although not part of the blind public key proposal, I would suggest devices could emulate other devices which are likely to have built-in drivers. For example, a USB device could emulate a memory device; although writing to special "files" on the device might result in side effects such as the file data being encrypted when read back.

This may not be so much of a problem, if two-part devices are used. The owner has a portable token, maybe like an electronic car key on their keyring, which uses wireless technology to link to a USB receiver. USB Bluetooth or 802.11 WLAN receivers are already available.

Whatever device is used, it will be necessary for the user to authenticate themselves to the device when the private key is used. Again, it is likely that several mechanisms will be deployed. The simplest may be one or two push buttons, as are found on electronic car keys. For most people,

the convenience of a simple device outweighs the possible extra security. And that is a reasonable risk judgement; why should a bank account, with typical funds of $1000-2000, need greater protection than the keys to a car worth several times that amount?

But it is entirely feasible to have stronger authentication mechanisms. A biometric finger-print swipe reader would seem an obvious choice. Note the ownership of the device – it belongs to the customer, just like the rest of their physical keys. They have an incentive to protect it. And any biometric information remains on the device, the property of the owner – so there is no risk of the data being misused elsewhere. This will help overcome any personal or legal reluctance to use biometric technology.

Commercial aspects

For this to be successful, the commercial aspects also need to be right. As with most developments, there are winners and losers. The main potential losers are the public CAs, as there becomes less need to buy expensive certificates. However there are still huge opportunities in closely-related fields. Rather than have extremely limited liability for who keys are signed, the future CAs can provide high-quality reference checking and referral services. There may be no need to sign a server certificate; but there will be a need to verify an organisation is good to do business with, possibly backed up by escrow services.

Hardware tokens can be funded by the consumer initially, helped with some "encouragement" by financial organisations to protect their keys securely. Because they are customer devices, there is no contention on which organisation should provide them or brand them with its logo – so the obstacles of consumers requiring an unacceptably large number of devices need not arise. A single device is adequate; preferably a backup device, and maybe others if strict segregation of credentials is required.

Getting browser modifications is perhaps harder, but the changes tend to be within the scope of plug-ins or cryptographic APIs, so can be created by third parties.

So in conclusion, there are no commercial show-stoppers to the blind public key proposal; indeed, it opens up new markets for common authentication stores and services.

Practical case

My employers, Dresdner Kleinwort Wasserstein (DrKW), have been using a similar public key system for over five years. This authenticates aound 25000 employees and customers to a several hundred web-based applications. Not all the details are identical, as the original design started from a different set of requirements, but there is great similarity, sufficient to prove that the principles work in real life.

The DrKW certificates do contain a small amount of personal information, though that is more for administrative convenience. They don't have any inherent authorisation properties, which are instead looked up in the directory. So an employee may have two different client certificates, one on a desktop and one on a laptop. They provide access to those applications they are authorised to use. Changing authorisation is simply a matter of updating a central directory entry.

If they think that a certificate may have been compromised in any way, or when the expiry date of a certificate is reached, it's a quick process to request a new certificate, replacing the old one. This can be done in under a minute, as a self-service process.

How well does blind public key meet the requirements

The requirements listed at the start are all well met:-

Public key technology (not just blind public key) meet the non-replayable, mutual authentication, end-to-end authentication, no re-used secrets/disclosure proof requirements. Replacing CRLs with a white-list approach meets the scalable requirement. The ability to use different credentials under different rules permits a proportionate approach and permits mobility of access. Hardware tokens extend end-to-end authentication to include the human user. The requirement for verified bilateral trust does not devolve trust to authorities, gives people power to make their own decisions, but also allows a "web of trust" to be constructed to aid such decisions. And there is a progressive route towards the new technology, without requiring large-scale replacement.

Kim Cameron's "seven laws of identity" are also well met:-

1. **User Control and Consent**: Digital identity systems must only reveal information identifying a user with the user's consent. Blink public key meets this by only revealing an anonymous identifier.

2. **Limited Disclosure for Limited Use**: The solution which discloses the least identifying information and best limits its use is the most stable, long-term solution. Similarly the least possible identifying information is disclosed, and that information is not secret.

3. **The Law of Fewest Parties**: Digital identity systems must limit disclosure of identifying information to parties having a necessary and justifiable place in a given identity relationship. The blink public key component of an identity system limits disclosure to the bare minimum as required. (Other identity information may be held by organisations and might be disclosed, so it's not sufficient to check just the blind public key component.)

4. **Directed Identity**: A universal identity metasystem must support both "omnidirectional" identifiers for use by public entities and "unidirectional" identifiers for private entities, thus facilitating discovery while preventing unnecessary release of correlation handles. Blind public key supports omnidirectional (shared) and unidirectional (multiple credential) identifiers, depending how the user wishes to trade simplicity against privacy.

5. **Pluralism of Operators and Technologies**: A universal identity metasystem must channel and enable the interworking of multiple identity technologies run by multiple identity providers. Blind public key tokens as described above support multiple technologies. The server components may need testing, but there is no inherent reason for lock-in to any technology.

6. **Human Integration**: A unifying identity metasystem must define the human user as a component integrated through protected and unambiguous human-machine communications. Blind public key recognises the human is part of the whole communications process. Use of a hardware token with user controls protects that part of the communications. Dialogues should be designed for unambiguous communication, though this is outside the scope of the current paper.

7. **Consistent Experience Across Contexts**: A unifying identity metasystem must provide a simple consistent experience while enabling separation of contexts through multiple operators and technologies. Blind public key enables as consistent an experience as possible, allowing different technologies, devices, providers to be used.

In summary, blind public key meets or exceeds the requirements for many types of identity systems for e-business.