October 2010

Blue Waters OTP Decision Guide

Blue Waters Security Team

This whitepaper lays out the requirements for a two-factor authentication system for Blue Waters and evaluates several solutions with respect to the requirements. The appendix also provides the reasons for employing two-factor authentication on Blue Waters in the first place.

National Center for Supercomputing Applications

Table of Contents

1. Introduction

The Blue Waters security plans and proposal identify the use of two-factor authentication for all interactive login sessions, in particular, by means of one-time password (OTP) tokens. This document's focus is then to inform the decision of a specific OTP solution. The higher-level justification for the requirement of OTP on Blue Waters is deferred to Appendix A.

While NCSA already utilizes a small Cryptocard OTP service in-house, it is no longer covered under a maintenance contract and the hardware needs to be retired soon. Also, the admins of this system have not been satisfied with its management capabilities. Therefore, with contracts expired and before a major expansion of OTP use at NCSA, we are taking the time to evaluate alternatives. Applying the lessons learned from administrating this system and taking the advice of users regarding our original OTP solution at NCSA, we have developed a new set of requirements and re-evaluated current market solutions—of which there are many more choices today.

In this whitepaper, we first discuss the requirements that were developed, and next we consider how each vendor meets, or fails to meet, our needs. We also consider developing our own OTP solution based on open standards and open source software components, describing what would need to be done to implement that solution.

2. Requirements & Desired Capabilities

These requirements are derived from the experience NCSA's security team has gained with the current Cryptocard infrastructure. Requirements were further refined based upon informal discussions[1] with users and system administrators (both within NCSA and at other HPC centers) our desire for future interoperability with other HPC sites, and usability concerns.

2.1 Physical Token (Mandatory)

While use of a physical, as opposed to a virtual, token is a required for any solution, it is not requirement for *every* user to have a hard token. Still, any solution must support the option of both "hard" and "soft" tokens, preferably with the option for a single user to have both.

It is desirable that the chosen solution supports as wide a variety of hardware tokens as possible. Many of our potential users already have one or more hardware tokens in their possession. If a user's time-based[2] token vendor is supported by our system and they are willing to share their token key, this single token could be used to authenticate against multiple OTP servers. This is in fact how VeriSign supports

[1] No formal survey was done, but we did talk to a variety of stakeholders, including PRAC team members and OTP administrators at other HPC sites.

[2] Event-based tokens cannot be shared across multiple sites unless the event counter is shared as well.

Paypal, eBay and many other sites with a single token. Having fewer tokens reduces confusion and inconvenience for users as they can avoid the need to carry a hardware token for every site they need to access.

Of course, one drawback of physical token sharing (as described above) is that the user (and security administrators at the other site) must trust that NCSA will protect the privacy of their token IDs. NCSA would be a trusted third party with access to a shared secret. However, in the light of cross-site projects (e.g., XSEDE), user authentication across multiple sites already requires at significant degree of trust between sites[3].

2.2 Smartphone Tokens or SMS (Mandatory)

Users have expressed a strong desire for smartphone soft tokens, since they already carry their phones at all times and would therefore not be required to carry any additional hardware. While not as secure as a separate hardware device, the increased risk is nominal compared to the gain in usability. Therefore, we have made the support of soft tokens for major smartphone OSes (i.e., IOS, Android and Blackberry) or support for sending codes via SMS[4] a requirement for any solution.

We do not want to use email gateways for one-time passwords due to the insecure nature of email transport of passwords. Furthermore, the Blue Waters Acceptable Use Policy will prohibit use of software-based tokens on a laptop or desktop system, due to the ease with which such systems can be compromised[5]. If we can technically prohibit soft tokens on a desktop we will, but we recognize this is often not possible and must address it in policy as well. Finally, native smartphone applications are preferred, as it is unrealistic to expect users to first install a separate runtime environment for just the token (e.g., Java ME).

2.3 OATH Tokens (Desired)

Using an open standard for both hardware and software tokens would give us the most flexibility to add pre-owned user tokens to our database or integrate with other institutions more readily. Therefore, we desire the use of OATH for any OTP solution. Additionally, a large number of supported tokens tend to drive the token market downward, decreasing cost.

There is also an additional benefit of choice when purchasing new tokens as a user would not be limited to the style of token selected by the majority. Still, this is a small benefit, as most users will choose the "free" tokens Blue Waters will provide as opposed to purchasing their own.

A drawback of this increased flexibility is the need for a method for users to securely provide us with an existing open authentication (OATH) token key. Also, while we

---

[3] Users often install public/private key pairs at one site that can be used to gain access to their account at other sites.

[4] SMS introduces a reliance on cell networks, which is a quality of service concern.

[5] In the case of SMS, it will have to be a real mobile number and not a Google Voice number that relays to an email account.

would prohibit the use of software tokens residing on a user's workstation or laptop[6] via User Agreements, there would be no concrete way for us to stop a user from providing us a token key generated by such software.

2.4 Pre-generated Password List (Desired)

There should be a method to send users a short, pre-generated list of OTP passwords for the following two reasons. First, some new accounts may need access faster than a physical token can be mailed out to the user. For example, if we have a verified fax number, a list of one-use passwords could be generated and faxed to the user. Second, there will inevitably be cases when a user may misplace, lose, or forget their hardware. In this case, a user who has already established a secure communication method (e.g., by providing a fax number registered with their account) could be sent a list of one-use passwords to use until a replacement token is sent.

2.5 Multiple Tokens (Mandatory)

A user must be allowed to have multiple, simultaneously valid tokens associated with their account (e.g., a hard token and a soft token on their phone). This would give users even more token flexibility without adding undo administrative overhead. Note that an NCSA-issued hard token would be needed to bootstrap the process of registering a soft token or federating with another OTP system anyway—since our proposed means of identity vetting is to mail the hard token and have its receipt notarized. Furthermore, every user should have a hardware token as a back-up even if it is not their primary OTP generation method.

2.6 Radius Authentication (Desired)

For cross-site interoperability, a RADIUS server should provide OTP authentication. The RADIUS server would handle two-factor authentication on its backend while providing a broadly supported authentication service. Any software that supports Pluggable Authentication Modules (PAM) can utilize the RADIUS protocol. In addition, almost all modern network equipment supports RADIUS natively. RADIUS is also the likely point of integration if we federate with any other sites in the future.

2.7 Concatenated Passwords (Mandatory)

It is important to only provide users with a single password prompt. If the second factor were not a PIN on the device, then the user would be required to concatenate their OTP response and second factor password into a single string. This removes the necessity for challenge-response protocols and leaves us with a very straightforward method of RADIUS authentication. Applications and services that wish to authenticate against our OTP infrastructure would not have to support challenge-response or multiple password prompts, thus significantly increasing the likelihood that we could integrate OTP authentication into some future, never-before-considered service or web portal.

---

[6] Remote system compromise was a strong motivating factor for implementing OTP. A software token residing on the user's workstation can be stolen almost as easily as a plain user password.

2.7.1 Kerberos Second Factor (Strongly Desired)

We currently plan to use Kerberos as our other factor for authentication. NCSA has been using Kerberos for a very long time, has a strong understanding of its inner workings, and our Kerberos infrastructure will remain in place for the foreseeable future irrespective of Blue Waters.

Many OTP solutions query an SQL database or LDAP directory for a user's password hash comparison. This method will not work with Kerberos as the encrypted form of the user's key is not known to the system. Therefore, a requirement to use Kerberos is that the OTP (or backend RADIUS) server must be able to extract the user's password from their concatenated response and attempt authentication using the Kerberos protocol[7].

2.8 Locally Managed (Mandatory)

We must have full control and management rights over our authentication systems. There should be no reliance on external vendor systems. While we may interoperate with other sites, we must still be able to stand alone and operate with tokens we have issued ourselves. This does not preclude the ability to interoperate with other sites through the sharing of users' time-base token keys or an externally facing RADIUS service.

An internal OTP authentication system can be more easily adapted to provide authentication to private LANs if we are not relying on services external to our network.

2.9 Quality Support (Mandatory)

We intend to develop in-house expertise for any OTP setup we use. However, there are likely to be occasions when we need to rely on the vendor or request a feature/bug fix. It is imperative that vendor support is fast, reliable and comprehensive.

3. Vendor Selection

A significant number of OTP vendors were initially considered. We discuss the vendors below and how they met or fell short of our requirements. We halted our investigation of a vendor as soon as we realized that they failed to meet one of the hard requirements, and thus we have few details on some products. The results are summarized in Table 1.

3.1 Cryptocard

NCSA has the most experience with Cryptocard OTP as we've operated a small installation for several years. Cryptocard meets many of the hard requirements, but not most of the desired features. OATH tokens are not supported in favor of proprietary tokens that require proprietary hardware to program. While multiple simulta-

[7] Kerberos pre-authentication must also be supported.

neous tokens are supported, they must all be Cryptocard products. Finally, concatenated passwords are not natively supported.

Lastly, our system administrators are unhappy with the lack of command line tools to add new users or RADIUS clients.

3.2 Nordic Edge OTP

Nordic Edge OTP is built upon OATH standards using a JAVA environment. Nordic Edge supports both HOTP and TOTP standards, and they have also built a fairly nice smartphone application called Pledge.

Initially, the Nordic Edge solution looked suitable, and their support staff was very responsive. However, while they unofficially claim to support SQL databases in addition to LDAP, they are heavily biased towards LDAP and have not clearly demonstrated an ability to make their product work with SQL.

They are currently unwilling to add multiple token support, and their OTP server does not support a concatenated Kerberos password as the second factor. The Nordic Edge server queries LDAP for an encrypted password rather than relying on LDAP to perform authentication. While it is not a hard requirement to support Kerberos as a second factor, concatenation and multiple token support is mandatory.

3.3 RCDevs OTP

RCDevs is another vendor based on OATH standards, which inherits all of the hardware and software token benefits that OATH provides. RCDevs also supports Mobile OTP (MOTP) and Yubikey tokens. However, the only backend database they support is LDAP. While their implementation provides the interface to use Kerberos as a second factor (through LDAP), NCSA does not currently have extensive experience managing an LDAP directory.

RCDevs informally claims a willingness to work with us to add additional functionality that we desire, such as multiple valid tokens, but it's unclear how that relationship would work or if it would cost extra.

3.4 RSA SecurID

RSA SecurID is a proprietary OTP solution with the largest market share, including some other HPC sites. SecurID does not support OATH standards and does not support using Kerberos as a second factor, instead requiring a user-entered PIN.

During the evaluation we found the GUI management interface to be a bit slow and non-intuitive. We also found it frustrating to be tied to Oracle for a database backend. Lastly, RSA is the most expensive OTP solution we investigated.

3.5 VeriSign

VeriSign Identity Protection (VIP) Authentication Service is a major player in today's secure authentication market. Historically, VeriSign was a driving force behind OATH as a competitor to RSA, but they've recently partnered to add support for SecurID tokens to VIP in addition to their support of OATH. Unfortunately, VeriSign

only appears to offer cloud-based authentication with no solution for a locally managed service.

3.6 Other Vendors

We looked at a number of other options, which were eliminated early in the evaluation process. These included Aladdin eToken (proprietary tokens, Windows-based), Mobile OTP (smartphone only), Wikid (challenge-response, not OATH), ID Control (blackbox solution) and DynaPass (Windows only, cloud-based). Other notable vendors are Priva Technologies and SolidPass.

Priva Technologies provides an entire security platform, but we have significant concerns about interoperability with other solutions. Its biometric requirements eliminate token flexibility and reduce usability, but we include it here primarily for completeness since it may be a viable solution for other systems or sites.

On the surface, SolidPass OTP looked like it might be a good candidate. They support both hardware and software OATH tokens and a wide variety of smartphones. However, all attempts to contact them with questions failed. Inability to contact their sales representatives with multiple attempts via both phone and email does not reflect well for their ongoing support services.

A hardware token vendor worth special mentioning is Yubico. They offer the Yubikey, which is a USB based token that eases OTP use by automatically filling in password fields. It's gaining popularity and can be supported as a token option for users of an OATH-based system.

	Cryptocard	Nordic Edge	RCDevs	RSA SecurID	Verisign
Physical Token	**Y**	**Y**	**Y**	**Y**	**Y**
Smartphone	**Y**	**Y**	**Y**	**Y**	**Y**
OATH	N	Y	Y	N	Y
Pre-gen List	Y	Y	Y	Y	Y
Multiple tokens	**Y[1]**	**N[2]**	**Y[2,3]**	**Y[1]**	**n/a**
RADIUS	Y	Y	Y	Y	n/a
OTP+Pass	**N**	**N**	**Y**	**Y**	**n/a**
Kerberos	N	N	Y[4]	N	n/a
Local Server	**Y**	**Y**	**Y**	**Y**	**N**
Quality Support	**N**	**Y**	**Y**	**Y**	**N**

Table 1: Vendors and Requirements (hard requirements in blue)

1. Only proprietary vendor tokens supported.
2. Multiple, simultaneous TOTP tokens will work if they use the same token key.
3. RCDevs is willing to add multi-token support in a future release.
4. Kerberos is supported if your LDAP infrastructure uses it on the backend.

4. In-House Development

An alternate option that would meet every requirement (and all the desired options) would be to develop our own OTP service in-house, based upon open authentication standards.

On the one hand we would have complete control over the entire solution, but the flipside to this is the lack of vendor support. Rather than licensing fees for OTP server software we would be spending time and money developing and maintaining some additional infrastructure. However, this is not without precedent at NCSA. In fact, such an OTP solution should contain fewer lines of code and less maintenance than that which we already maintain with our many patches for OpenSSH. It would also require far less customization than all the customized and hand-written Bro policy-analyzers.

An internally developed OTP service would consist of several components.

4.1 RADIUS Server

As the service interface to authentication clients, the RADIUS component is a key element that would act as the backbone of the entire solution. There are numerous free implementations available that can be adapted to our needs. It is within the RADIUS server that we would split the user-provided concatenated password, handing off one piece for Kerberos authentication and the other for verification against the OTP component. RADIUS can also provide integration with other sites if we open it up to external queries.

4.2 Kerberos Integration

Many Blue Waters users will already have accounts either in the NCSA or TeraGrid/XD Kerberos realms. Building our own service would provide the oppor-

tunity to support multiple Kerberos realms that would be configurable on a per-user basis. Any given BW or NCSA account name could be mapped to a different username and/or Kerberos realm for second factor authentication. If we choose to trust external Kerberos realms, those could be configured as well.

4.3 OTP

The one-time password portions would be based on open standards provided by the Open Authentication community. Much of the code for supporting event-based (HOTP[8,9]) and time-based (TOTP[10]) OATH tokens is already written and freely available. The support of OATH would provide users with a wide range of supported hardware tokens as well as freely available smartphone applications. In addition to OATH, we may be able to add native Yubikey and/or Mobile OTP support.

We would have full control over the inner workings to add security features to prevent replay attacks and create custom timeouts when brute-force attempts are detected. Support of multiple, valid tokens should be trivial.

4.4 Database

User token keys and their associated information must be stored in some form of database. Typical vendor solutions rely on an existing LDAP or SQL database; so this is a component that we would likely have to build and manage regardless. The database would store user account information, token keys, token event counts (where applicable), as well as replay and brute-force attack data.

The database is key to providing a redundant and reliable service that spans multiple servers, as it is the source of data replication.

4.5 Management Interface

There must be a method for administrators at NCSA to manage user OTP keys in order to help debug and solve authentication issues. Also, security personnel need a way to disable accounts. It is expected that user accounts would be automatically added and deleted, but any OTP infrastructure requires human involvement to handle physical token management.

4.6 User Managed Key Interface

For both an in-house solution as well as vendors supporting OATH, we have to consider methods of obtaining keys for user-owned tokens. OATH token keys consist of a long character string, which creates ample opportunity for miscommunication via phone, or fax.

To minimize this, it may be prudent to create an interface where users can manage their own OTP token keys (or at least register new tokens). A user would be required to login with a valid OTP device, as well as their second factor, to verify their identity. In this scenario, a user would need their initial hardware token or, at a

[8] http://tools.ietf.org/html/rfc4226
[9] http://www.nongnu.org/hotp-toolkit
[10] http://www.ietf.org/id/draft-mraihi-totp-timebased-06.txt

minimum, a pre-generated list of valid passwords to register a new token. A table of free OATH smartphone applications is provided in Appendix B.

5. Recommendations and Cost Comparison

Only three choices meet our hard requirements. They are RCDevs OTP, RSA SecurID or an in-house solution. Vendor pricing models vary based on the number of users, so for a middle of the road comparison we look at prices for supporting 1200 concurrent users, which is more than we expect for Blue Waters.

	RCDevs	**RSA**	**In-House**
Server Hardware	$ 10,000	$ 10,000	$ 10,000
Initial Development	.1 FTE[1]	$ 0	.3 F TE[6]
1st year license	$ 14,340	$ 48,108	$ 0
1st year tokens	$ 24,000[2]	$ 101,736	$ 24,000[2]
Support (years 2-6) [4]	$ 71,700	$ 37,100	.1 FTE / year
Token Renewal	$ 24,000[3]	$ 101,736[3]	$ 24,000[3]
Administration[5]	.5 FTE / year	.5 FTE / year	.5 FTE / year
TOTAL	$144K + .1 FTE + 0.5 FTE / year	$299K + 0.5 FTE / year	$58K + .3 FTE + 0.6 FTE / year

Table 2: Cost Comparison for 1200 Users

1. Some development of the backend database and User Managed Keys interface is required.
2. Based on an average cost of $20 per hardware token (often cheaper) assuming we buy 1,000 initially. We also assume the use of free OATH smartphone applications.
3. RSA tokens are based on a 3-year token life and must be re-purchased for both hardware and software tokens. For RCDevs and In-House OATH tokens, we assume a similar turnover of users and/or battery life.
4. This is the cost of licensing/support across all of those years, not a per-year cost.
5. Administration is a combination of server and software maintenance as well as user token and account management. These tasks are likely split among several people and/or groups.
6. See Appendix C for details of this work.

6. Recommendation

Two vendors meet all of our hard requirements: RSA and RCDevs. However, RSA does not have the strongly desired capability of Kerberos support for a second factor, and RCDevs still requires some additional development work (In fact, perhaps as much as doing it completely in-house). Neither of these solutions meets all of our desired features, though RCDevs should (with some extra work) in a future release when it supports multiple tokens.

As a result, we recommend moving forward with in-house development of an OATH-based OTP service, **if we can acquire supplemental work for Blue Waters to do this**. We could even make available our solution to any other NSF communities that are interested (e.g., XD). Of course, we would not be providing official support to those sites unless NSF chose to fund such activities. It is also low risk within the current timeframe as we could deploy the RCDevs solution quickly if things go awry.

The in-house solution provides us with the most flexibility as well as the deepest understanding of the internal workings of the final solution. There is an added level of comfort and trust knowing exactly how the system works. Funds set aside to pay vendor license and support fees are either saved or converted into staffing dollars leading to a lower total cost of ownership.

We recognize that there are two risks major risks to the in-house development. First, there are our assumptions about the initial and on going development costs. While we have dived deeply into these investigations and believe that it should be simply a matter of connecting together the appropriate open source components, our estimates could be wrong and we would have no support contracts with vendors to help us with any issues. We would rely on the expertise we develop in-house and some free and open source projects.

Second, we assume the continuation of an on-going Kerberos infrastructure at NCSA. While it is difficult to imagine it going away any time soon, with all the internal dependencies on Kerberos, it is still an assumption. If it ever does go away, then Blue Waters would need to continue the Kerberos realm on its own to support its users or we would have switch mechanisms for the second factor of authentication.

With this said, **the low risk option is clearly RSA**. They are the most established vendor and are unlikely to disappear, they provide good support, and they cut off the need for us running any sort of Kerberos realm or other service for the second factor. The downside of course is we cannot control PIN strength, and they are many times more expensive than the alternatives.

Glossary

Event-based OTP – An event-based one-time password relies on the shared token key and a count representing the number of passwords already generated using this token. Each use of a password increments the count on the server so it knows which password to expect next. To help guard against the token being out of sync with the server, the server will typically accept a configurable number of passwords that are next on the list.

HOTP – Specific event-based OTP implementation as defined by OATH and RFC 4226

MOTP – OTP implementation used by Mobile OTP.

OATH – Initiative for Open Authentication, http://www.openauthentication.org

Time-based OTP – A time-based one-time password relies on time synchronization between the token and server. The current time is used in conjunction with a shared token key to generate a password. Typically a given password is valid in a

shifting time window on the order of 30 or 60 seconds. The OTP server provides a check to prevent against replay attacks.

Token key – A unique hardware or software token key shared between the user's device/software and OTP server, used in the various algorithms to generate a one-time password in conjunction with an event counter or the current time.

TOTP – Specific time-based OTP implementation as defined by OATH

Appendix A: Justification for OTP

In recent years there has been a steady increase of instances of user credential harvesting on clusters in HPC environments. In the last two years, 80% of the incidents on HPC systems at NCSA (which accounts for 50% an FTE year for investigations and a week of system downtime) have been the result of user credential compromise.[11] Because this is our primary threat, and it appears this trend will continue for some time to come, we need to develop a new authentication system for Blue Waters with strong mechanisms to mitigate the impact of credential theft.

Traditional user authentication mechanisms have been passwords (Kerberos), SSH public/private key pairs, and PKI certificates. All of these methods suffer from a common weakness, that the user is primarily responsible for the safe keeping of authentication credentials. Users need to choose good passwords, and if they are using public keys or certificates they need to apply strong passphrases to protect those credentials (many employ none). Since users access our systems from remote hosts out of our control, their credentials are stored or entered on untrusted hosts. Far too often these hosts are infected by malware that harvest unencrypted keys or passwords as they are entered. These credentials, once harvested, can be reused by an attacker to access our systems. The attacker then often waits for the next zero-day kernel exploit to log in to our systems as a valid user and utilizes this new exploit to gain root access more quickly than we can react. This puts us in a constant state of reaction, cleaning up after attackers as new privilege escalation exploits come out.

An effective way to break this cycle, which we have used for administrative accounts for years, is to make the credential something of little value for the thief. One Time Password (OTP) systems use a rolling code so that each access requires a different credential that cannot be reused. In this way, there is no longer a benefit to stealing a credential and holding onto it.[12] An attacker could still, in theory, hijack the user's existing connections to create another SSH channel if they control the user's machine, but this is something they could do now anyway. The reality is that it requires

---

[11] Since we need OTP for system administrators anyway, calculating the benefit for user tokens should compare these numbers versus the incremental cost of 1130 additional OTP tokens and their maintenance plus the hard to quantify notion of usability.

[12] Though, it is in theory still possible to steal the token key from a soft token. This is where hard tokens really shine, and why we are against using any sort of soft token on a laptop or desktop.

a more sophisticated attacker, and the attacker loses the ability to choose the time and source from which he would access our systems as he would need to tag along when the user logs in. So it is not a kind of attack that can propagate rapidly across user accounts, giving us much more time to react. Even for those adversaries with the skill to mount an SSH hijacking attack, most would likely move on to other sites not using OTP because it is not only easier for someone just looking for vulnerable hosts, but there is more value in a credential that can be used from anywhere, anytime.

This is another step in the arms race between attacker and protector. Unfortunately, it is a game we are obliged to play. We need to raise the bar if the cost (including user aggravation) is low. This will give us significant benefit until the day that the majority of sites raise the bar to the same level and force the attackers to do more work for less gain.

We believe that this is a cost worth bearing for two reasons. First, we have a moderately sized user community (less than 1130 users) for Blue Waters, and tokens have become quite cheap. Second, OTP technology has advanced significantly, eliminating the need to carry many large cards or FOBs. For example, several vendors make products that allow the use of either a tiny hardware device or a soft token, which is software you install on your smartphone. We have recommended this more flexible type of OTP system over the one currently used at NCSA.

To help reduce usability concerns, the BW authentication model will also allow users to obtain short-lived x.509 certificates via MyProxy using two-factor authentication. These certificates will provide the user with the ability to authenticate to multiple hosts, transfer files via GridFTP, and communicate more easily with partner sites while minimizing how often they need to use their OTP token.

Below we highlight the specific pros and cons of OTP use for Blue Waters users.

Pros of OTP

As described above, the majority of incidents currently experienced at NCSA arise from credential harvesting. Since OTP directly addresses that kind of threat, we can significantly reduce the number of incidents with this one change. This will reduce the amount of time spent in reactionary cleanup, decrease our response time, and give the security team more time to explore novel and proactive approaches to security. In the end, users will see an increased uptime and system availability.

Second, the security plans for Blue Waters depend on specific threat models, and OTP has been a working assumption that influenced many other decisions. For example, by raising the bar for the type of adversary that can get onto the login node, we lowered the risk of allowing users interactive logins to their compute nodes (through the login node). So the inconvenience of a token is balanced with some additional usability.

Lastly, deploying this authentication service allows us to easily achieve a NIST 800-63 Level of Assurance 3 in the future, assuming we restrict access to FIPS 140-2 Level 1 validated tokens. This is a level above most educational institutions and allows other organizations with higher levels of trust (e.g., DOE labs) to more readily share resources and federate with us. Of course, this comes with a price we discuss in the next section, namely that there will be fewer sites whose certificates we readily trust.

Cons of OTP

The largest complaint from users is likely to result from a requirement to carry multiple physical tokens that they can lose or forget. This usability issue is in part addressed by our requirement for soft token support for smartphones. Further, other HPC sites that have started rolling-out OTP solutions are eager to federate with us, including the XSEDE team. Such federation is simple through federating RADIUS servers. Therefore, a user in the HPC community with one OTP token will likely be able to login to many other sites with this token, thus reducing the likelihood of a keychain full of fobs.

We will only be able to accept certificates from CAs that require OTP to get a short-term certificate. If XD does not operate as planned with an OTP-enabled CA, that will mean we will accept few external certificates on day one. However, for services like GridFTP, we only need one CA to issue certificates trusted by both ends of the connection. Therefore, as long as we get our Blue Waters CA TAGPMA-approved, most other sites should have no issue trusting our CA (even if we do not trust theirs because of the lower level of assurance for authentication). We will just need to educate Blue Waters users that such services should use the Blue Waters issued certificate.

Account creation will be slowed by the requirement to mail physical tokens, though federating with other HPC OTP systems or providing a short list of one-time-passwords to use upon account creation could mitigate this issue. If we do not already have a trusted communication channel with a new user (in most cases we will not), then we would mail account details and require a notarized receipt of the packet in any case. Therefore, adding an OTP token to the package would not slow the process in most cases.

Finally, OTP is a modest, but additional, cost. There is the cost of running and managing an additional service, the cost of the tokens, potentially the cost of software licensing fees, and some increase in help desk time dealing with lost tokens. However, other sites have found that a small replacement fee for tokens greatly reduces the number of lost tokens. This is not extremely expensive, due to the limited size of the Blue Waters user community, and we believe that we more than make up for it by eliminating the costs, both explicit and implicit, associated with the majority of incidents we currently face. However, there is always the risk that reality does not match theory, and either the cost-savings could be reduced or the management costs increased.

Appendix B: OATH Smartphone Applications

RCDevs maintains a good website listing **free** OATH smartphone applications[13]. We provide a summary table below.

Application/Vendor	iPhone	Android	Black-berry	Java-Phone	Windows Mobile	HOTP	TOTP
Google Authenticator	X	X	X			X	X
OATH Brand Token	X					X	X
iOATH Brand Token	X					X	
DS3 Oath Token	X			X		X	
Pledge Token	X	X	X	X	X	X	
Android Token		X				X	X

Appendix C: Initial Development Breakdown for In-House OTP

OTP Authentication Service

In-house development of an OTP solution will require piecing together multiple components. By following the path of user authentication we start to see how the parts may fit together and get an idea of the level of development required.

When a user supplies their password, it's handed off by the authentication client to RADIUS. There are multiple open source RADIUS implementations available, one of which, FreeRADIUS, has support for authentication modules that could be leveraged. Regardless it will have to be modified to deal with the user supplied concatenated password. The OTP response portion of the password could be a variable length of 6-10 characters. This variability adds some minor difficulty in splitting the OTP response from the user's Kerberos password. One straightforward, but inefficient method would be to attempt OTP authentication against all five possible password lengths. Alternatively, we can include the expected response length in the database for each token key a user possesses.

This opens up the topic of the database requirement. A database of some kind is needed to store user token keys and Kerberos realm preferences. A minimal schema is provided in Appendix D for reference. The initial database query is used to determine if the user's account is enabled and that they have at least one valid token key.

Once the user's password response is split into its OTP and Kerberos components, authentication takes two independent paths. These can be performed serially or in parallel. The user's enabled token key information as well as their Kerberos username/realm preferences can also be included in the first database query if that turns out to be more efficient.

---

[13] http://www.rcdevs.com/tokens/?type=software

The OATH algorithms for validating an OTP response are publically available. It's unclear if it's best to pull in the algorithms and integrate them as necessary or start with something like otpd (formerly of Tri-D Systems, now a Google Project). We would initially focus on the HOTP and TOTP methods, but keep the door open for MOTP and Yubikey at a later date. After successful authentication, several variables have to be updated in the database to keep track of HOTP counters and prevent replay attacks.

The second factor of authentication will be the user's Kerberos password. By this point we will have already queried the database to obtain the user's Kerberos principal and realm. While we would support multiple realms for flexibility and convenience to the user, only those realms, which we trust, will be allowed. Factors to consider are whether or not remote realms support pre-authentication and have upgraded beyond single DES keys. Kerberos authentication is simplified since we aren't concerned with storing the user credentials and should be fairly straightforward.

Generally speaking, we can handle redundancy and synchronization using a dual-master MySQL setup. Another option for synchronization would be using the gsmd (global state manager) functionality from otpd. There are still some hurdles here to prevent replay attacks and this is the area that will be the most challenging and where we would need to be the most careful. Aside from replay attacks, we have to implement brute-force protection. As an example, Cryptocard will lock a user completely after 10 failed attempts. That's a poor solution given brute force attacks in the wild can (and have) frequently locked user accounts. A configurable timeout is probably better. The RADIUS server component can maintain some simple state for accounts that are temporarily blocked for a brute force timeout. This would reduce unnecessary database queries.

User Key Management

The purpose of user key management is to let the users manage their own OTP tokens and reduce administrative/helpdesk overhead. The interface would require them to first login with two factors using an OTP hardware token we provide or a pre-generated list of passwords.

The difficulty with token key management is the transfer of the HOTP/TOTP shared key. The token key is a long string, and it's easy to make mistakes. The user should be able to provide a key or we can generate a key and provide it to the user. Another possible source of confusion is that different OATH applications sometimes expect the token key in different formats. For instance, the "OATH Token" app expects a hex encoded key, while "Google Authenticator" expects a base32 encoded key. Google Authenticator (at least on the iPhone) also comes with a barcode scanner. We could provide a barcode with the token key on a website. However, most soft tokens would require the user to type in the token key.

The interface would need an immediate way to test the OTP token to verify it is working without requiring the second factor again.

It's not recommended to let users manage their own Kerberos user/realm pairs. We would want to choose which realms we trust and would have to manage the krb5.conf file to support those realms.

It doesn't appear that most HOTP/TOTP tokens support re-syncing based on a challenge-response. All of the smartphone tokens we've looked at don't have an option to enter a challenge. There's some talk of OATH Challenge Response Authentication (OCRA)[14], but it doesn't yet appear to be widely supported.

As an alternative to challenge-response, for a given HOTP key and a user provided OTP response, it wouldn't take long to brute force the current token count. In essence, the server would sync to whatever the token's count is, rather than the token being reprogrammed. Regardless, we need some method to fix this for HOTP tokens. For software tokens the best may be to login with the provided hardware token, then delete/re-add a token key or reset the 'count' value. This requires further investigation.

Token Administration and User Management

The management interface needs all of the functionality of the user key interface and more. While token keys need to be handled manually, user account information can be automatically added/removed as part of the NCSA account allocation process.

There needs to be an easy (possibly one-click) method to pre-generate a list of valid one-time passwords for a user that has lost their token or who needs immediate access while waiting for their hardware token in the mail. This method should automatically generate a valid token key to insert into the database. Pre-generated password lists should have an associated expiration date. The management interface must also have a method to temporarily (or permanently) disable a user's account at the behest of the NCSA Security Team.

[14] https://www3.tools.ietf.org/id/draft-mraihi-mutual-oath-hotp-variants-11.html

Appendix D: Minimal In-House SQL Schema

```
CREATE TABLE `users` (
 `username` varchar(255) NOT NULL,
 `brute` varchar(255) default NULL,
 `date_added` datetime NOT NULL,
 `date_removed` datetime default NULL,
 `enabled` tinyint(4) NOT NULL,
 `date_disabled` datetime NOT NULL,
 `reason` text,
 PRIMARY KEY  (`username`)
)

CREATE TABLE `tokens` (
 `tkey` varchar(255) NOT NULL,
 `username` varchar(255) NOT NULL,
 `enabled` tinyint(4) NOT NULL,
 `date_added` datetime NOT NULL,
 `date_removed` datetime default NULL,
 `type` enum('HOTP','TOTP','MOTP','Yubikey') NOT NULL,
 `length` tinyint(4) NOT NULL,
 `count` int(11) default NULL,
 `replay` varchar(255) default NULL,
 `expiration` datetime default NULL,
 `date_disabled` datetime default NULL,
 `reason` text,
 PRIMARY KEY  (`tkey`)
)

CREATE TABLE `realms` (
 `username` varchar(255) NOT NULL,
 `principal` varchar(255) NOT NULL,
 `realm` varchar(255) NOT NULL,
 KEY `user` (`username`)
)
```